Course on Graphical User Interfaces (GUI)

Session 1, part 1: Introduction, overview, teaching method

Gilles Venturini, venturini@univ-tours.fr, office 204/220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

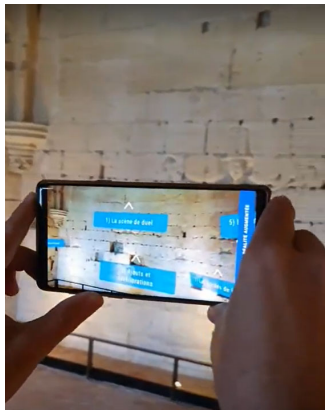
POLYTECH TOURS

université de TOURS

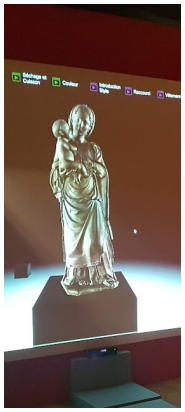
LIFAT EA 6300

Human-Computer Interaction, HCI (Interface Humain-Machine, IHM)

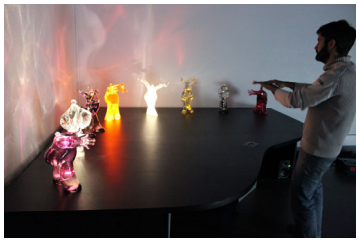
- A vast field of study …



Augmented Reality



Virtual Reality



Voice/gesture-based

Tangible Interfaces

Brain-computer Interfaces

Graphical User Interfaces

- Does not fit in a single slide!

Human-Computer Interaction => Graphical user interfaces

- This course will focus on:
  - Graphical user interfaces (GUI)
  - With standard input/output devices (mouse, keyboard, screen)
  - And standard widgets and controls: windows, menus, buttons, text boxes, etc
  - « WIMP »: windows, icons, menus, pointer

Graphical user interfaces

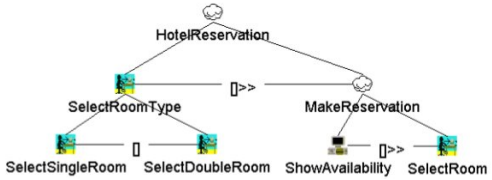
- What you will learn in 10 hours:
  - How to collect users needs
  - How to specify/model/design a GUI, following conception rules
  - How to implement and test a simple GUI with Python/Tkinter (not the best choice because no graphical editor, but no other possibility for this course)

Teaching method

- Inverted class:
  - No more lectures in large groups -> work in small groups
  - Course divided in 5 x 2H sessions
  - For each session: work at home + hands-on exercises in class
- For a 2H session in class:
  - you have work to do at home (from 10 to 30 min, depending on your level)
  - Work at home to be done before the class session (watch videos, read the course, answer some quizzes)
  - In class, hands-on work
- Evaluation
  - Participation to quizzes
  - Presence in class
  - Work in class (or in other projects)

How to produce a GUI?

1. Collecting/modeling the needs:
   - What problem are users solving (or willing to solve)? (entities, input/output, models)
   - Who are the users? (profiles or personae)
   - What tasks do they perform to solve this problem (or should perform)? (scenarios, use cases, workflow)



2. Creating:
   - Brain storming to discuss several alternatives (your GUI with pencil & paper)
   - Make a more advanced proposal (wireframe conception)
   - Prototype (partial or complete implementation of GUI functionalities, testable)

A sketch of the interface

Interface of a proposed system

https://www.tutorialspoint.com/human_computer_interface/interactive_system_design.htm

3. Implementation (consider technological constraints)
4. Evaluating the GUI (or part of)
   - Perform user evaluation => user feedback
5. Go back to 1 (do this cycle at least twice)

https://usabilitygeek.com/remote-usability-testing-best-practices/

Detailed content for each session

- Session 1: User needs
  - At home, watch « Session 1 » videos
  - In class: analyse a problem,
  - Outcome: personae, tasks models, other constraints and models, …
- Session 2: first GUI specs
  - At home: watch « Session 2 » videos
  - In class: brain storming, start designing
  - Outcome: pencil/paper specifications, mockups with graphical editor,
- Session 3: Design
  - At home: watch « Session 3 » videos
  - In class: continue design, evaluate design of others work, iterate
  - Outcome: final design/prototype

Detailed content for each session

- Session 4: prototype implementation
  - At home, watch « Session 4 » videos
  - In class: python/tkinter implementation,
  - Outcome: python code, executable
- Session 5: user evaluation, and iterate
  - At home: watch « Session 5 » videos
  - In class: perform tests of your GUI, user feedback, iterate
  - Outcome: evaluation protocol, test reports, final prototype

Course on Graphical User Interfaces (GUI)

Session 1, part 2: Collecting user needs

Gilles Venturini, venturini@univ-tours.fr, office 204, 220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

POLYTECH TOURS

université de TOURS

LIFAT EA 6300

Collecting the needs: problem modeling

- Consider the problem to solve
- And list all its entities:
  - That will play a role in the GUI
  - External entities: actors, systems, connected elements, inputs, outputs, …
  - Internal entities: components, inner (data) structures, modules, information flow, dependencies
  - And their relations

- Threats:
  - Be sure you are working on the right problem
  - Do not forget entities (or an important characteristic of an entity)

Problem modeling: an example

- Consider following problem: providing a GUI for booking a car
- External entities can be:
  - Car(s): model, energy, nb of seats (front, back), automatic/manual gearbox, …
  - Driver(s): name, age, driving license, …
  - Passenger(s): child, adult, …
  - Insurance: conditions, price, partial, full, …
  - Specific equipment: GPS, baby seat, …
  - Rental agency: location, …
- Internal entities can be:
  - List of cars (a table in a database, …)
  - Reservation: selected car, driver(s), passenger(s), from date, to date, pick up/drop location, …
  - List of reservations

Collecting the needs: user(s) modeling

- Consider the population of users for you GUI
- Typical information you need to collect:
  - Personal/cultural information: age, place of birth/living, gender, language, current level of studies or position or retired, skills (in Computer Science, in GUIs, typical software used, etc), typical device used (mobile, PC, Mac, …)
  - Role in the problem model
  - Tasks currently solved or to be solved (see the next step about tasks)
  - Expectations (with your GUI)

- Threat: wrong estimation of the real population of users => GUI useless!

How to collect information about users

- Introspection:
  - Pretend to be a user and think/behave as such
  - Threat: representative? You are just 1 over 8.000.000.000 … and you are the GUI creator
- Read documentations
  - About the context, users, problem
  - Threat: concepts are good but they are not real experiments
- Observation/interviews/immersion:
  - Conduct interview to know users, or watch them acting (solving the problem, possibly with think aloud protocol)
  - What user do, why they do it, how (what tools), when
  - What vocabulary they use, what errors they make, what sources of frustrations
  - Audio/video recording
  - Threat: time consuming, so small number of users (5, 10, 20, …)

How to collect information about users

- Forms (and remote testing):
  - Ask to answer questions in a form (like Google form),
  - Propose to use your GUI remotely, possibly record logs (but rare)
  - Good for obtaining many answers/comments
  - Threat: no control of the experiment, not as reliable and as rich as observing people
- Do not forget users with specific needs
  - Visually impaired people
- Remember that users might not know what is good for them

How to collect information about users

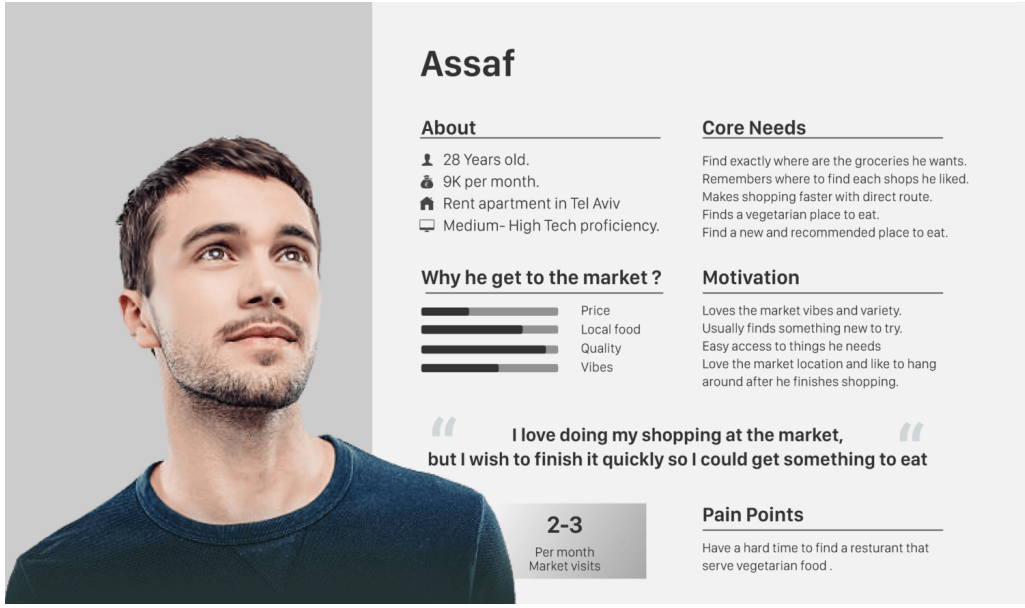
- Once the population of users is better known
  - Make groups of similar users (user segment), with estimated size in %

10% 60% 30%

  - For each group, create a profile/personae/empathy map (see next slide)
  - For each profile, study tasks/scenarios/use cases

Collecting the needs: user(s) modeling

- Make typical user models (profile or personae)
  - Bio, Personality, Frustration, Expectations, Goals, Motivation, Tasks, Devices, …



https://www.asi.fr/blog/pourquoi-creer-personae-est-utile-developpement-dun-produit-dun-service

https://www.justinmind.com/blog/user-persona-templates/

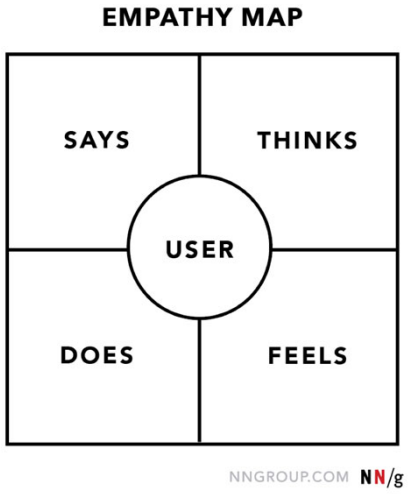
Collecting the needs: car rental example

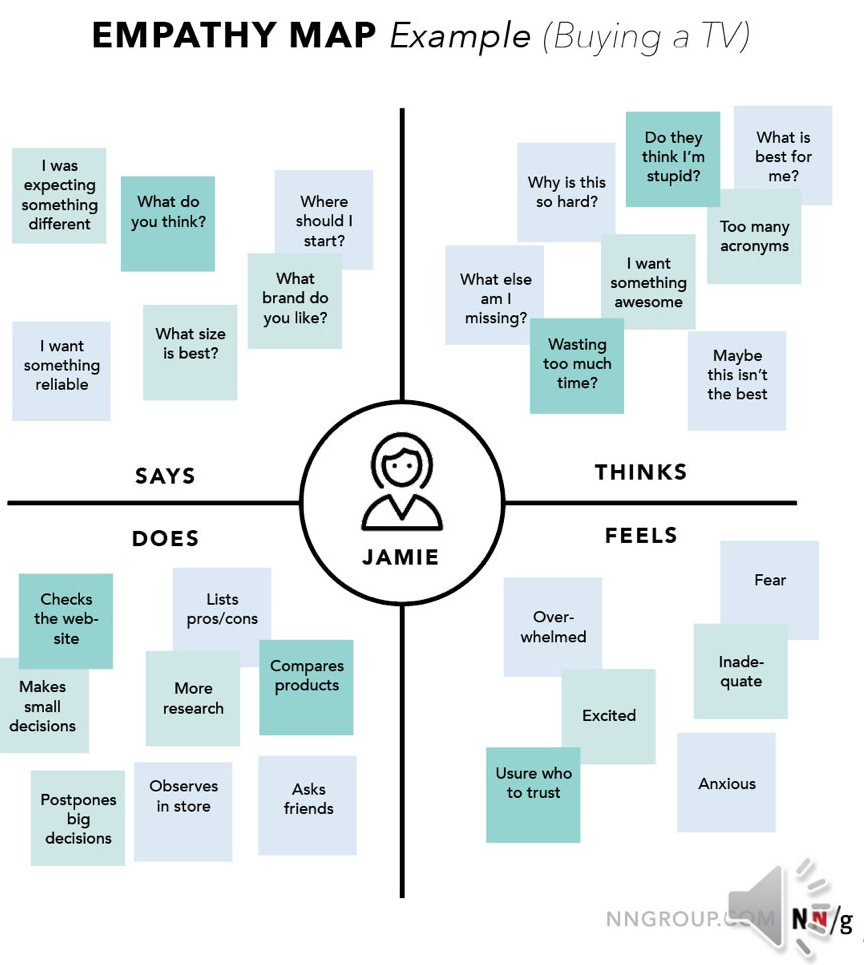
- Mary is 72 years old. She wants a comfortable car that respects the environment. She needs an automatic gearbox and parking system. She also wants many associated services. When searching for a car, the most important point for her is the day and time of pickup/drop. Usually, she uses the phone to make a reservation. With this new GUI, she prefers to use a laptop than a mobile, and she will need assistance.
- Matt is 24 years old. Overall, he really wants a sport car. He is even willing to adjust his dates of rental to get the car he wants. Red color is a must. He uses his mobile all the time. To get many “likes”, he wants to share the reservation of an awesome sport car on social networks.

- Segmentation: 70% of customers will be like Mary, and 30% like Matt.

Collecting the needs: user(s) modeling

- Empathy maps represent detailed information about several users with 4 categories: Says (interviews or task solving), Thinks (not said), Does (actions during task solving), Feels (emotions)
- Defined in brain storming session

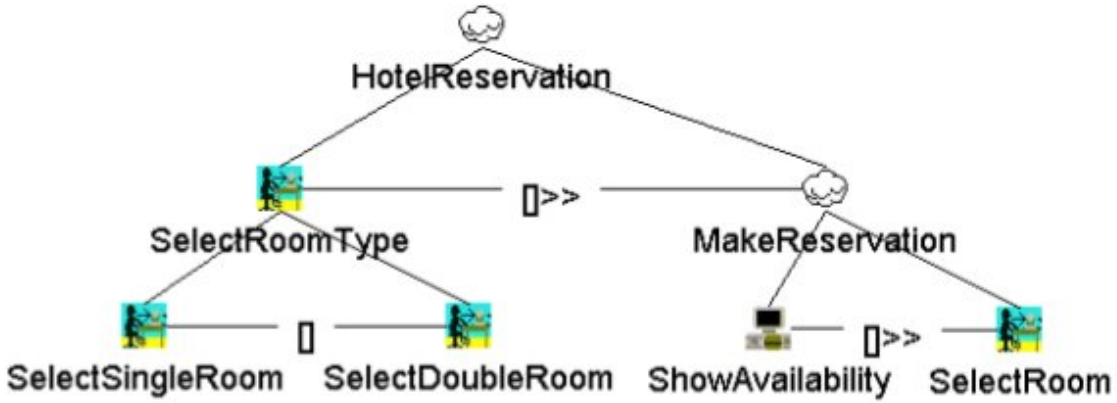




https://www.nngroup.com/articles/empathy-mapping/

Collecting the needs: tasks modeling

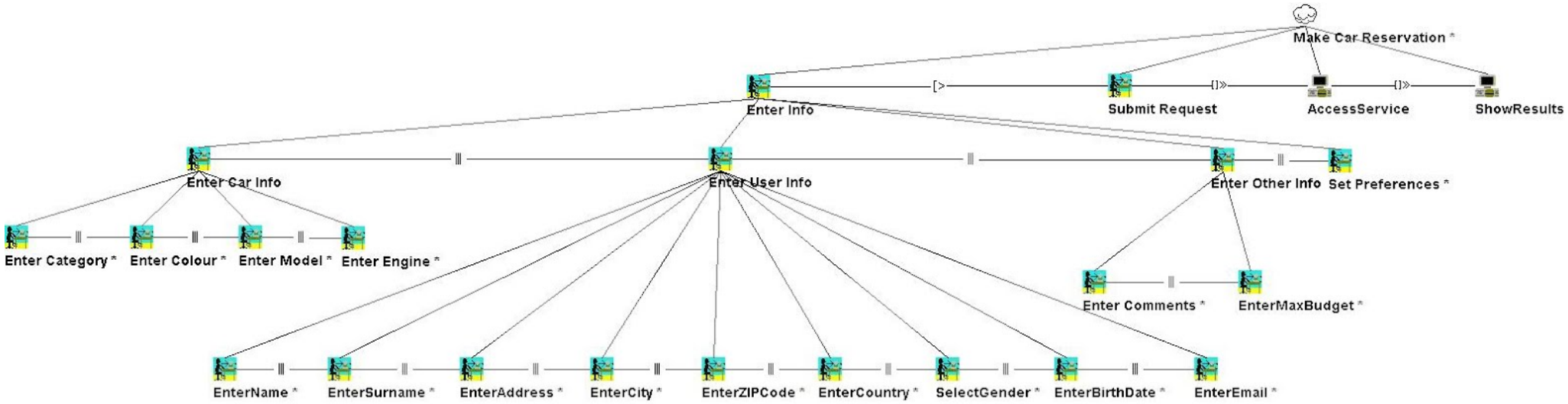
- For each user profile
- Consider the users goals/typical scenarios/use cases
- Describe each scenario with a list of tasks/actions that user must perform
- Describe the tasks sequence/dependencies
- Propose a hierarchical model (main task, sub-tasks, …)



https://www.w3.org/TR/task-models/

Collecting the needs: tasks modeling

- Suppose that Mary wants to book a car
- She'll have to do this:



https://www.w3.org/TR/task-models/

Collecting the needs: tasks modeling

- Tasks sequence: simple hierarchical representation

To book a car, Matt has to perform 5 tasks:

1. Specify the date and time, pickup/drop location, age of driver
2. Select a car among a list of results
    a) Explore the results
        I. Sort/filter the list
        II. Observe with details (color, horse power; gear box, parking system)
        III. Loop
    b) Select one result
3. Give additional information about drivers/passengers
4. Select special equipment
5. Pay

Use cases (in UML)

- See the Software Engineering course

Collect other important constraints

- Devices:
  - On which devices will you GUI run? Screen resolution? Computer power/memory? Touch screen?
  - Threat : GUI does not run well on target devices. GUI does not make use of possibilities offered by the device (screen resolution, touch screen)
- GUI software library
  - What graphical elements/widgets are available
  - Threat: library has too few elements (compared to your mockups)
- Project management:
  - Time/budget devoted to the GUI conception, implementation, testing
  - Threat: design takes a long time, so requires project time and budget
- Competitive analysis
  - Look at other GUI doing the same (good and bad ideas or design)

Exercise studied in class: myDinnerParty

- As part of social life, people organize meals with guests at home
- When? Who is invited? What food/drink to prepare/buy? What should guests bring? What activities?
- The goal is to propose a GUI for this app
  - Session 1: Collect user needs
  - Session 2: Create paper and pencil
  - Session 3: Mockups
  - Session 4: Prototype
  - Session 5: Evaluation

Exercise studied in class: myDinnerParty

For session 1:

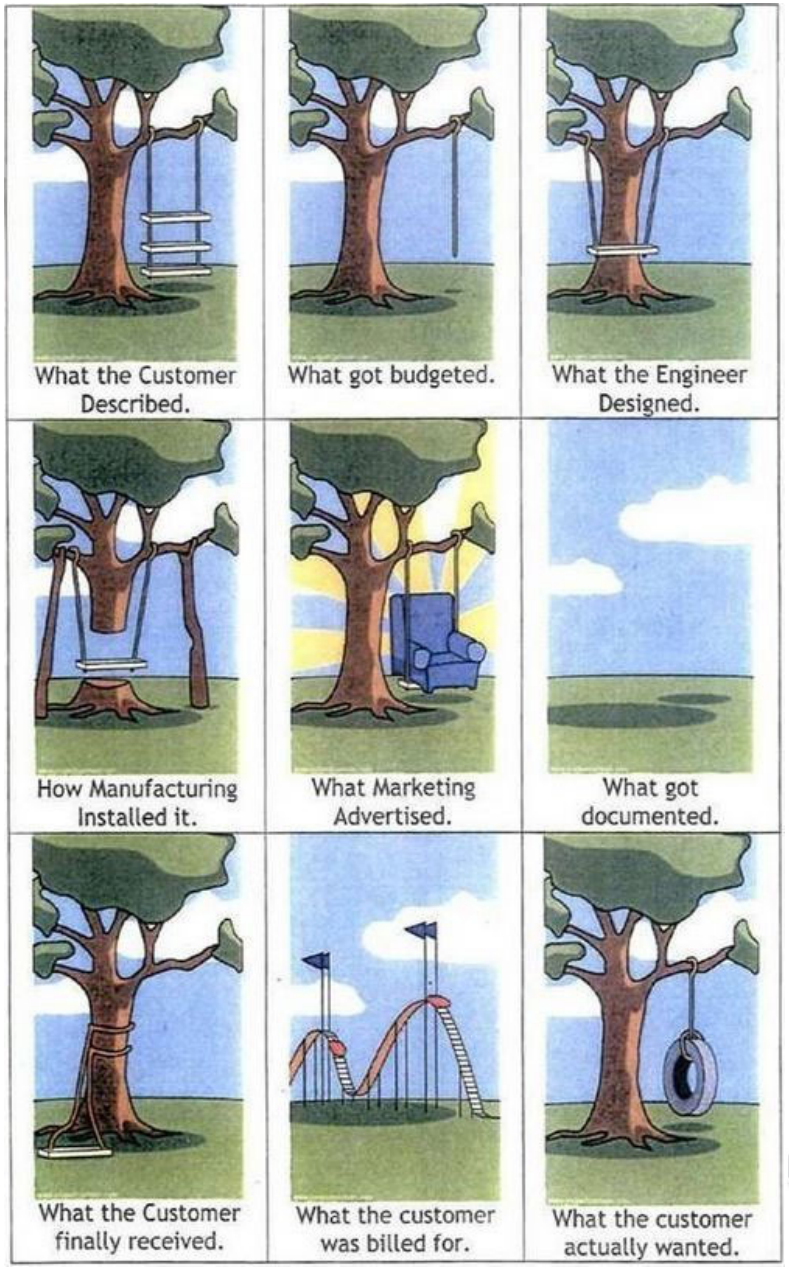
- Get user needs
- Perform a simulation in class with a group: one inviter, several invitees, the others take notes
- From a questionnaire

Outcome:

- Problem model, personnae, task models

Conclusion on this part

- Be sure to understand user needs!



Course on Graphical User Interfaces (GUI)

Session 2, part 1: Overall methodology for GUI design

Gilles Venturini, venturini@univ-tours.fr, office 204, 220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

POLYTECH TOURS

université de TOURS

LIFAT EA 6300

Creating the GUI: design methology

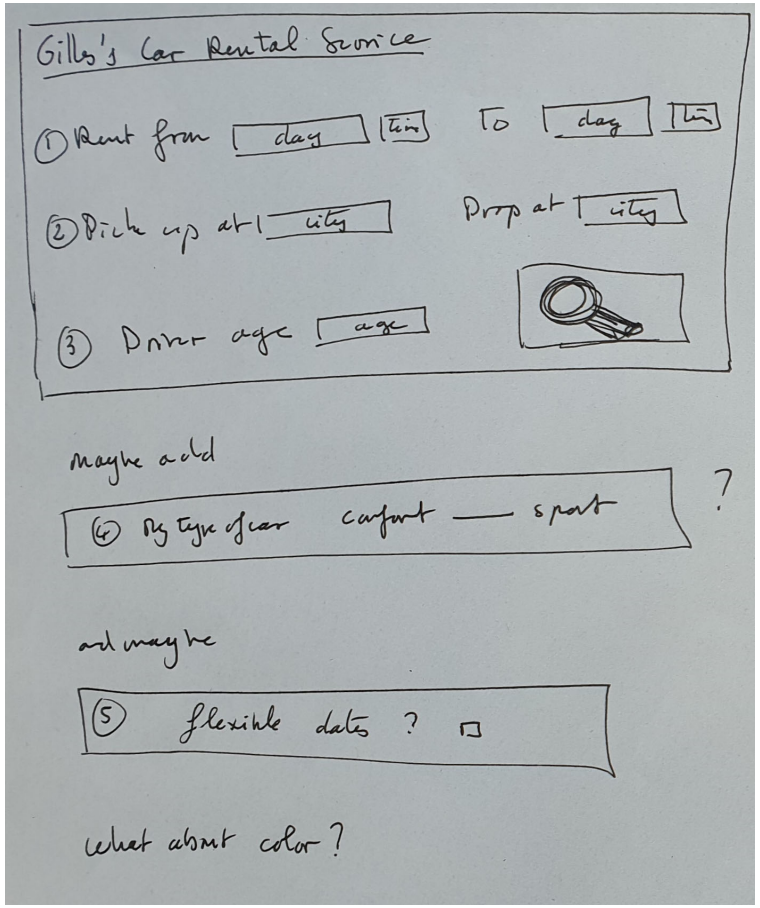
1. GUI with pencil & paper (sketch)
   - Make several proposals (approximate, draft)
   - Simulate user scenarios/tasks
   - Loop to adjust or select one and go to next step
2. GUI from wireframes to mockups
   - Select one design to improve, style, higher precision, yet no real interactions
   - Edit with software, present to users
   - Loop to adjust or go to next step
3. GUI prototype
   - Propose a partial/complete implementation, with interactions, dynamic behavior and realistic content
   - Test and evaluate with target users
   - Loop to adjust

Your GUI with pencil & paper

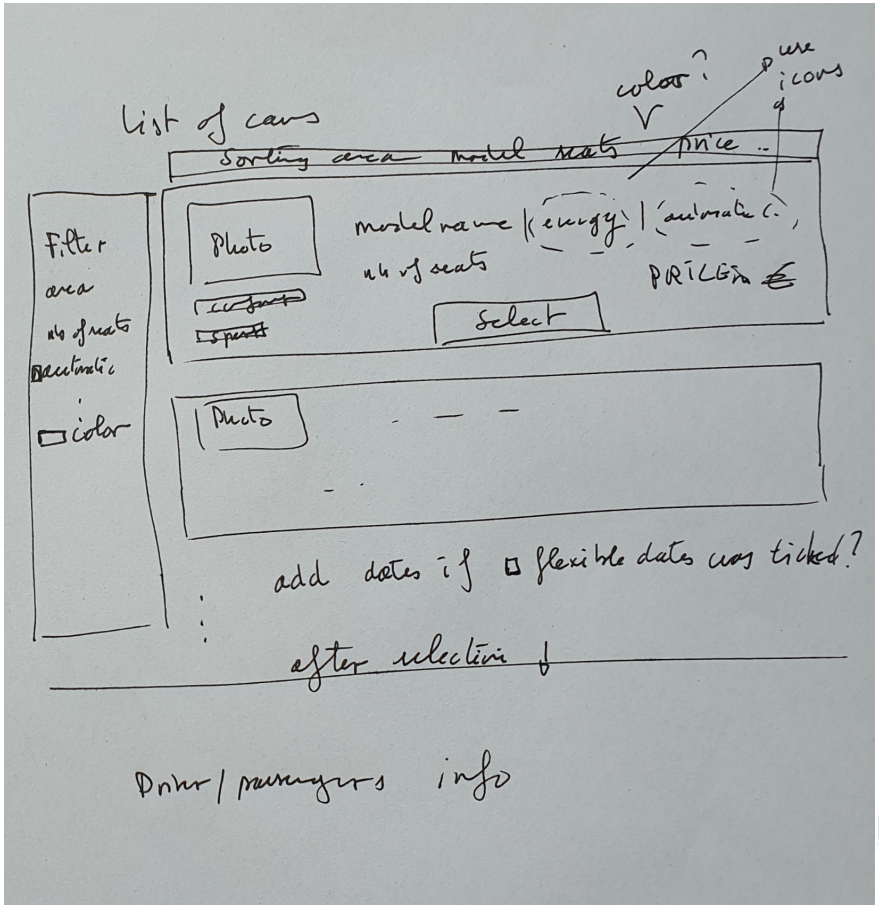
- Brain storming
- Make several proposals
- Design, layout, size of elements, style, etc can be approximate
- Simulate user scenarios/tasks: use several sheets of paper to render the dynamic behavior of the GUI (like windows opening, other changes, etc) and see if this fits the users tasks

GUI with pencil & paper: car rental example

Task 1 (Specify date, time, etc)

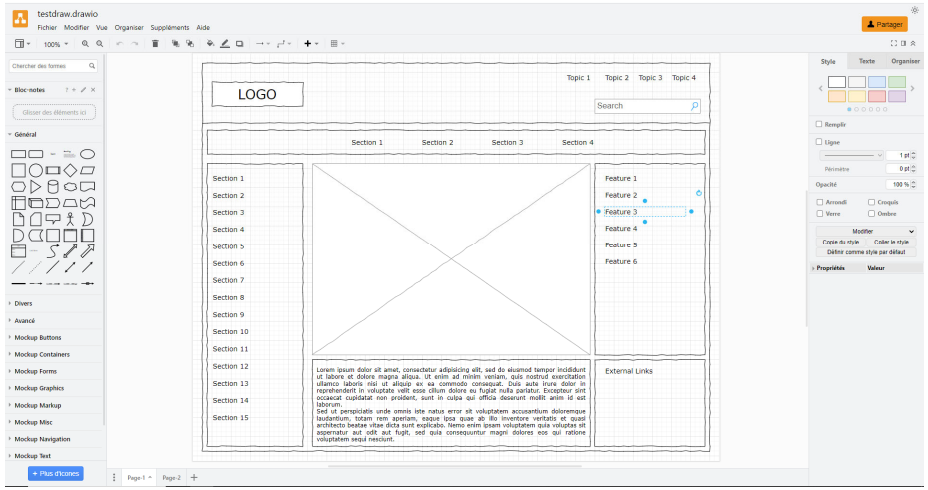


Task 2 (Select a car among a list of results)

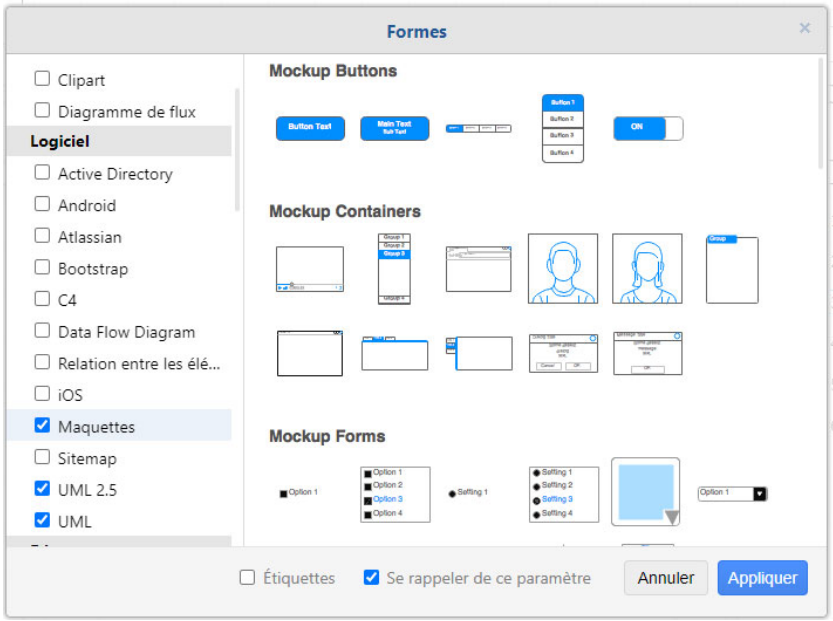


A more advanced proposal: mockups

- Work on one design from previous step
- Produce wire frame model, or mockups
- Elements properly placed, more details, static (some mockups software can support navigation)
- Simulate/check scenarios
- Can be presented to users (but requires explanations)



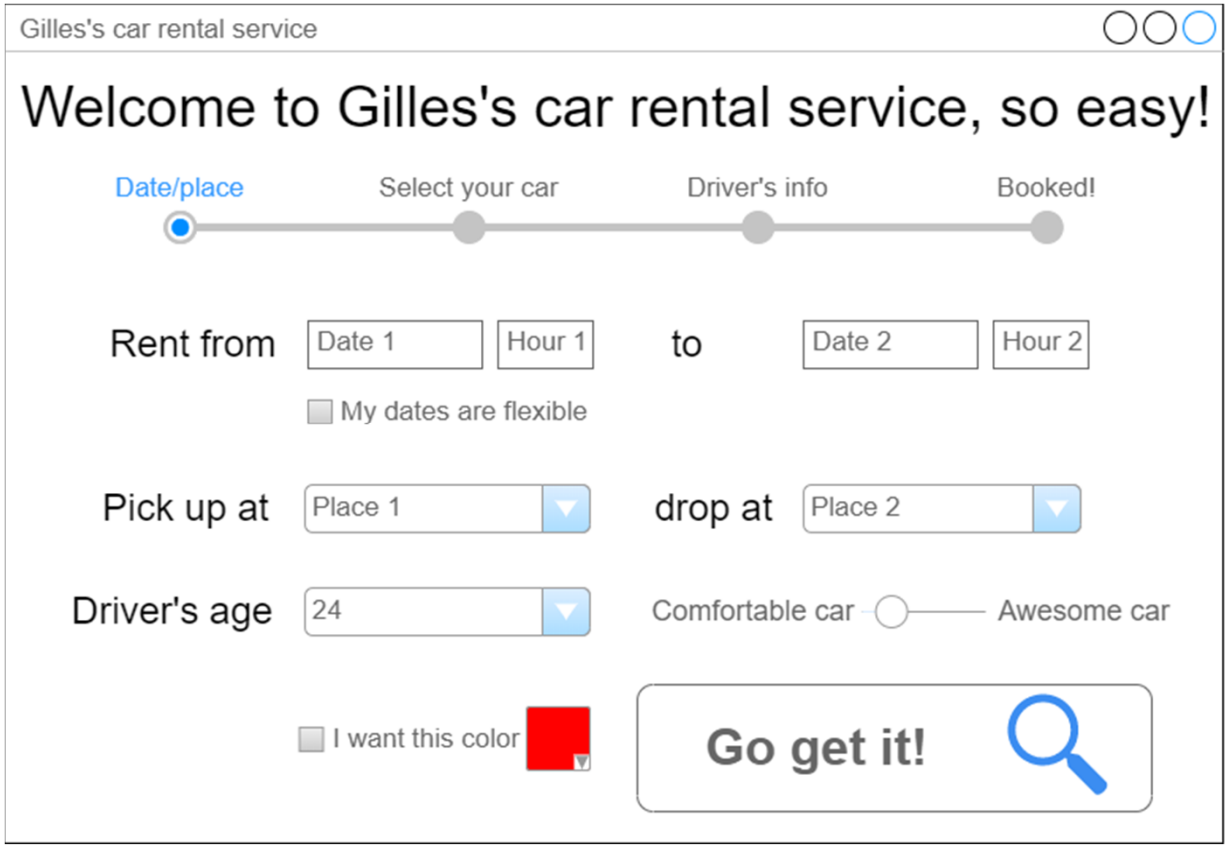
Software for mockups



- Draw.io (with specific icons/elements)
- http://pencil.evolus.vn/
- Figma, Balsamiq, Ninjamock
- https://moqups.com/ (has navigation)
- Threats: check that
  - Needed widgets/elements are present in editor
  - Widgets/elements used in your mockup can be implemented
  - Screen resolution is OK

Mockup example (with draw.io)

- Continuing our car rental simple GUI, mockup for Task 1



A testable prototype

- Last step: turn the mockup into a prototype
- Partial or complete
- Should be as close as possible to the final product (content, style, dynamic behavior, response time, etc)
- Must be evaluated by users (specifying which part to evaluate)
- Iterations are important

Course on Graphical User Interfaces (GUI)

Session 2, part 2: Finding a good design

Gilles Venturini, venturini@univ-tours.fr, office 204, 220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

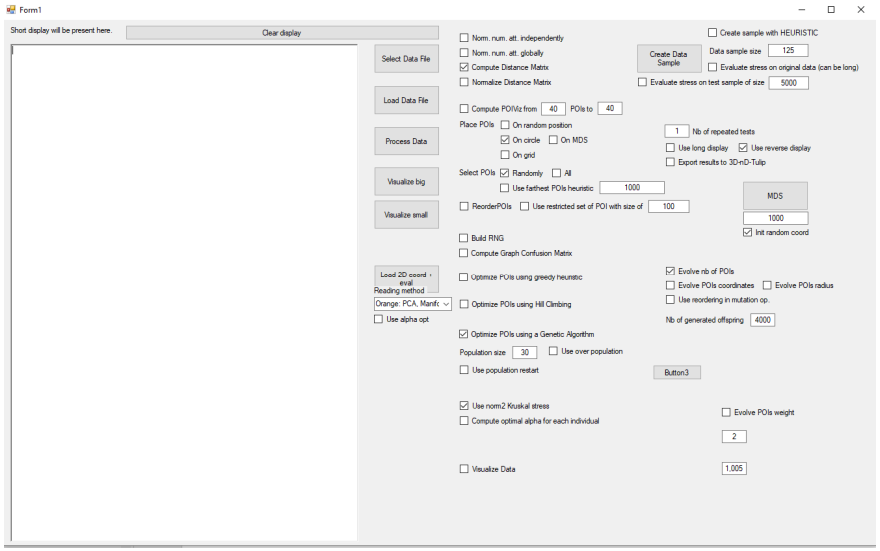
POLYTECH TOURS

université de TOURS

LIFAT EA 6300

GUI design is crucial

- Design includes: choice of elements, layout, style, interactions (dynamic behavior)
- Test this: https://userinyerface.com/
- Errors in design can have a very high cost (project failure, death of people in case of airplanes …)



GUI from a personal software (for research). Obviously, only me can use it!

- Assuming user needs are collected, you must also be aware of:
  - Human visual perception and information processing
  - GUI design rules (general, specific): visual elements, style, layout, interactions

Course on Graphical User Interfaces (GUI)

Session 2, part 3: Visual perception and human information processing

Gilles Venturini, venturini@univ-tours.fr, office 204

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

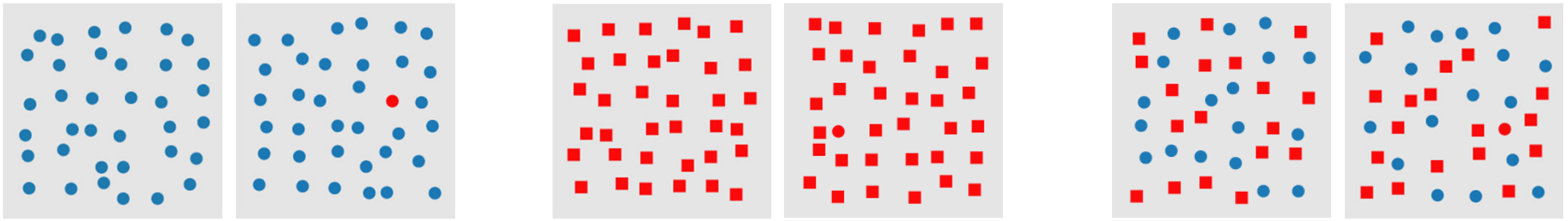
POLYTECH TOURS

université de TOURS

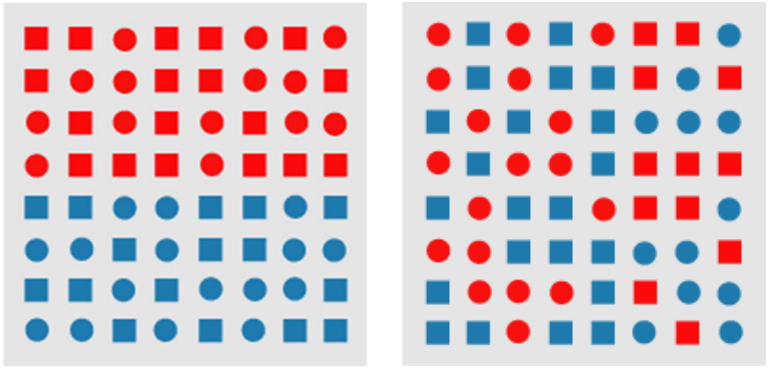
LIFAT EA 6300

Visual perception: preattentive perception

- Find an important information (red circle):



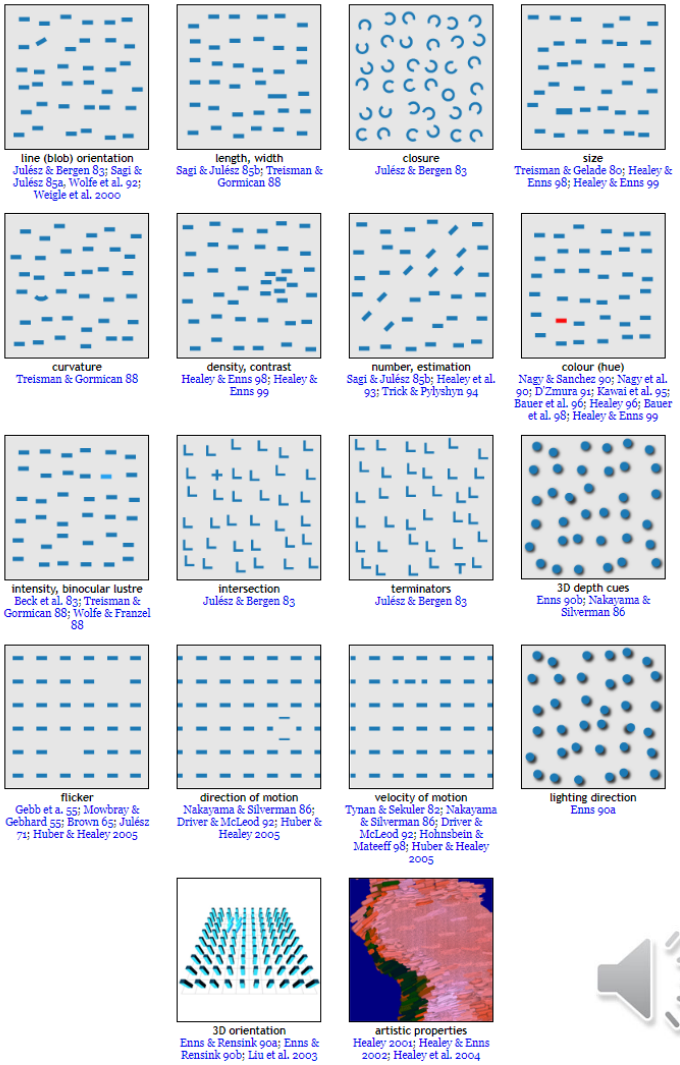
- 2nd example, important information = boundary



https://www.csc2.ncsu.edu/faculty/healey/PP/

Visual perception: preattentive perception

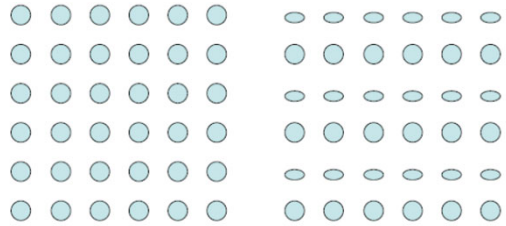
- Preattentive visual variables perceived in less than 250ms
- Information pops out
- To draw user's attention (critical situation) => use preattentive perception



https://www.csc2.ncsu.edu/faculty/healey/PP/

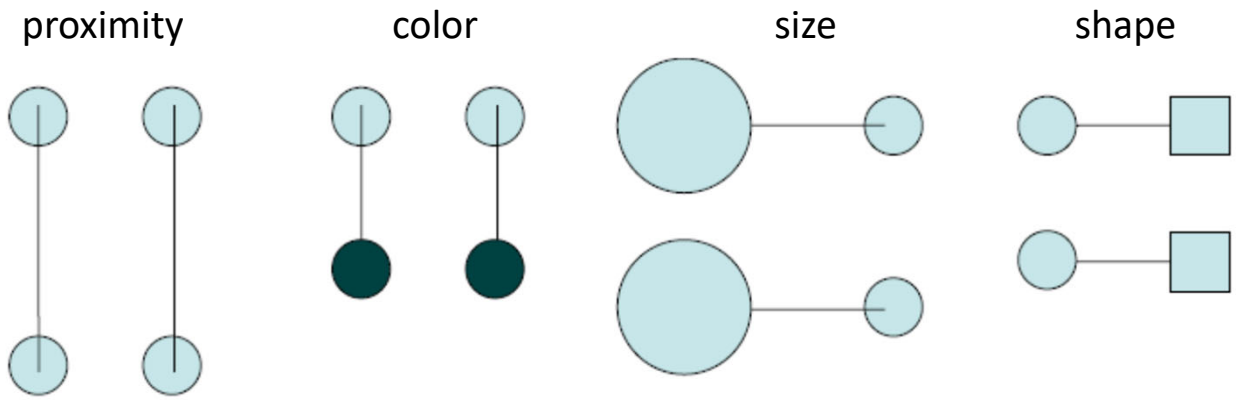
Visual perception: gestalt theory

- From simple elements to patterns
- Grouping by similarity:



Once you see a pattern, your brain will remember it.

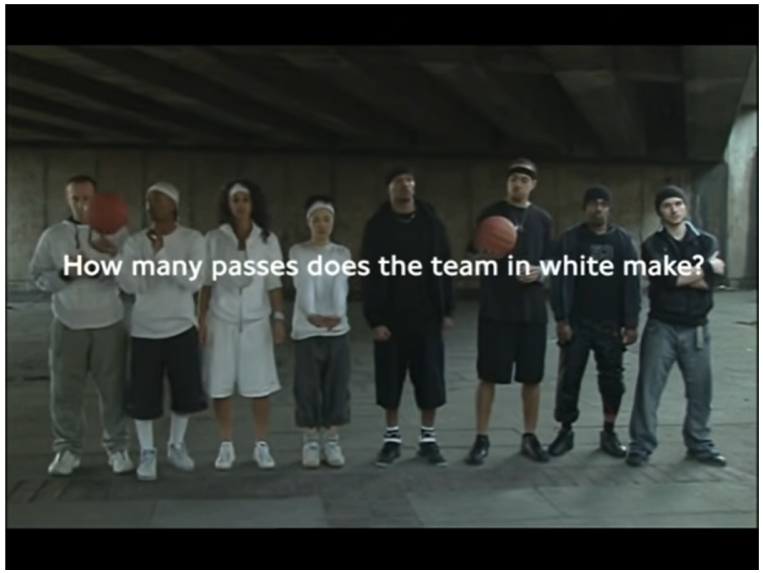
- A connector is stronger than:



https://www.usertesting.com/resources/topics/gestalt-principles

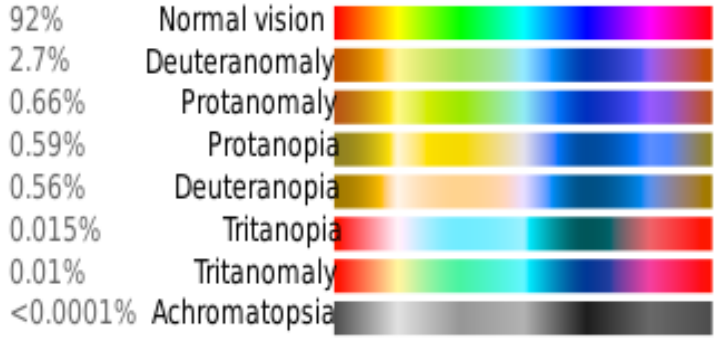
Visual perception: inattentional blindness

- When engaged in solving a problem, user can be “blind” to other elements (of your GUI):



- https://www.youtube.com/watch?v=Ahg6qcgoay4

Visual perception: colors



- In general, avoid using color (and gray scales)

- To distinguish symbolic elements (Home, Car, Work, Bank, …):
  - Use 12 colors at most red, green, yellow, blue, black, white, pink, cyan, gray, orange, brown, purple

- To distinguish numeric elements (Temperature, …):
  - Avoid rainbow scale: 
  - Try this site: https://colorbrewer2.org/


Order?

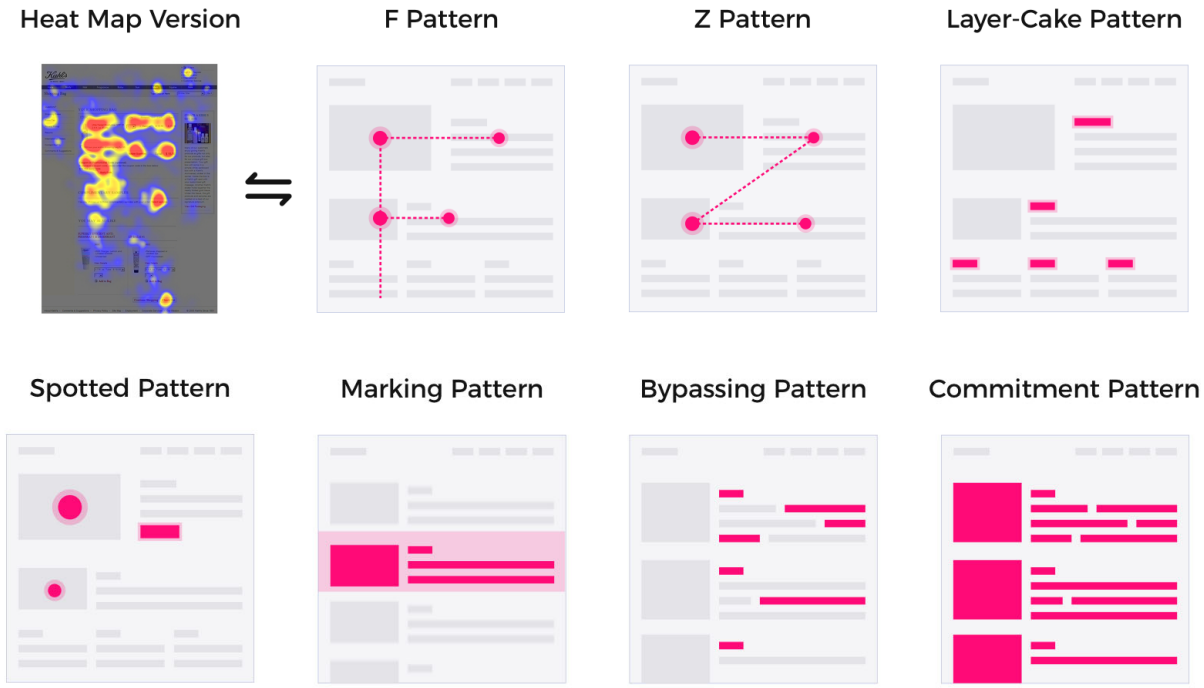
https://en.wikipedia.org/wiki/Color_blindness

Visual perception: scanning the screen

- Users scan the screen to get an overview (preattentive perception, pattern perception) and then concentrate on details (cognitive efforts)
- Visual hierarchy (from the overall picture -> details)
- Find a layout + design that:
  - Facilitates the GUI visual understanding and adoption
  - Drives the user attention where you want

Visual perception: scanning the screen

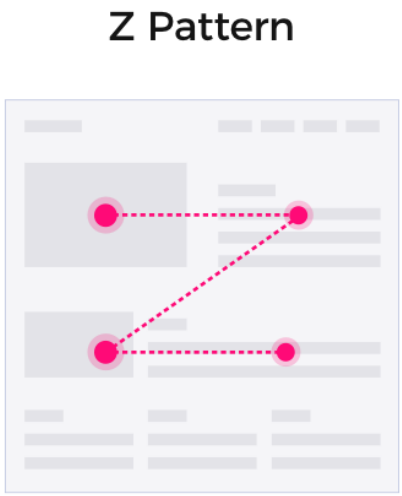
- Scanning patterns
- GUI (western countries):
  - left to right reading
  - Z pattern (possibly F)



https://www.toptal.com/designers/web/ui-design-best-practices
https://www.interaction-design.org/literature/topics/visual-hierarchy

Visual perception: scanning the screen

- Important elements at the top, left and then bottom of your GUI



https://www.toptal.com/designers/web/ui-design-best-practices

Visual perception: Recognition vs Recall

- Favor visual recognition rather than recall or cognitive load (example = printing a doc):

visual recognition

recall (what does this icon mean already?)

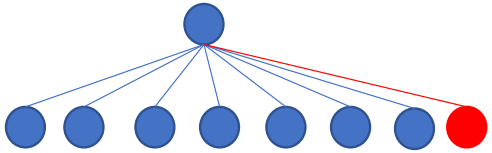
This part of the GUI is devoted to an important action: once your document is ready, you will find in this aera all the required options to print it.

cognitive load (reading long text, thinking)

https://www.toptal.com/designers/web/ui-design-best-practices

Human information processing

- Miller magic number: 7 +- 2 objects in human working/short term memory
- Consequence (avoid complexity >7):
  - Hierarchy: no more than 7 branches
  - Group: 7 +- 2 elements at most



- In a long list of elements, make chunks: 435754842 -> (435)(754)(842)

- Some exceptions (when memory is not concerned)

https://en.wikipedia.org/wiki/The_Magical_Number_Seven,_Plus_or_Minus_Two

Course on Graphical User Interfaces (GUI)

Session 2, part 4: GUI design principles

Gilles Venturini, venturini@univ-tours.fr, office 204, 220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

POLYTECH TOURS

université de TOURS

LIFAT EA 6300

Ben Shneiderman 8 Golden Rules of Interface Design

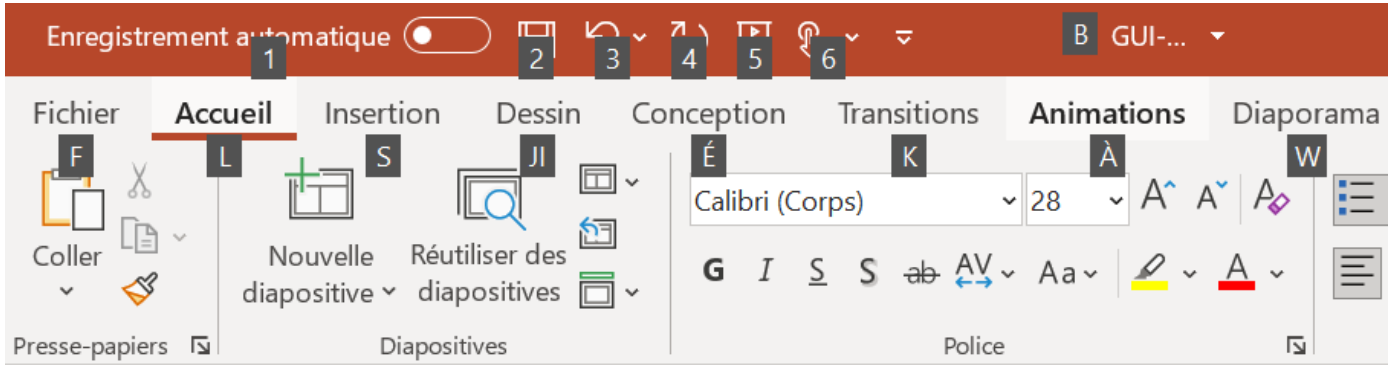
1 Consistency: homogeneous design, standards, similar elements in similar situations

- Choice of widgets, appearance, layout
- GUI behavior

2 Propose shortcuts to frequent users

- [Crtl]+[Alt]+[Del]
- [Crtl]+[c]

Shortcuts in Power Point

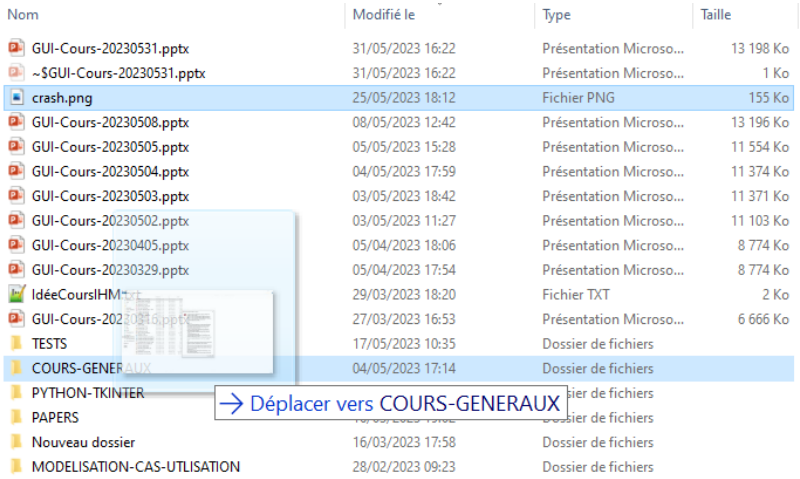


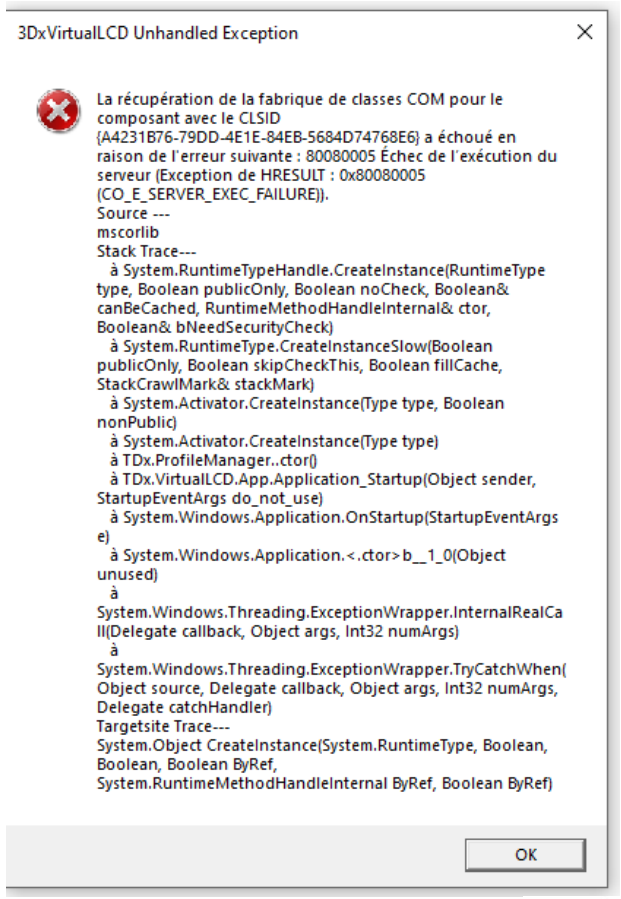
https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

Ben Shneiderman 8 Golden Rules of Interface Design

3 Offer informative feedback

- Clarify dialogue with users
- Always let the user know what the system is doing
- When the user is triggering/doing something in the GUI, the GUI must provide feedback (a kind of change, or information)





https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

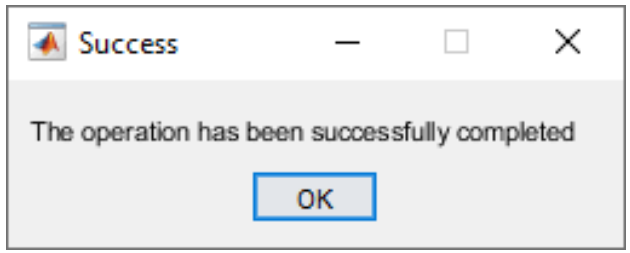
Ben Shneiderman 8 Golden Rules of Interface Design

4 Design dialogue to yield closure:

- Let user know when a task/action is finished

⇒ Use bread crumb/fil d'Ariane (see slides on this)

⇒ Or send message



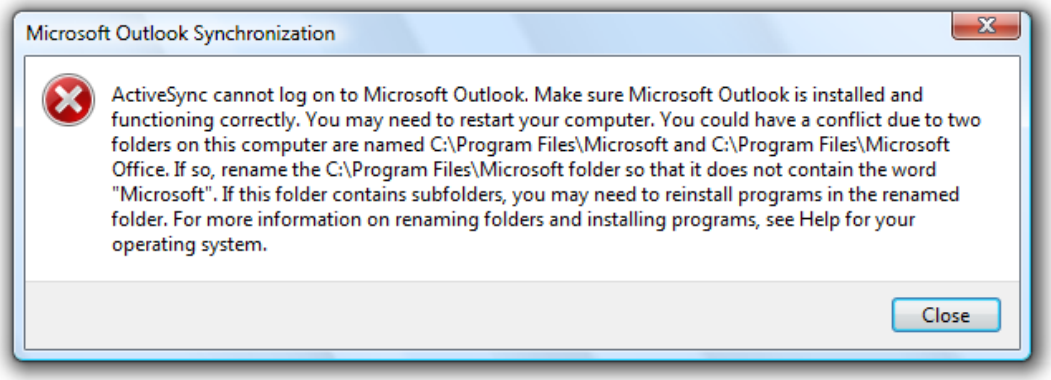
- Let user stop a current task without problems (data loss, etc)

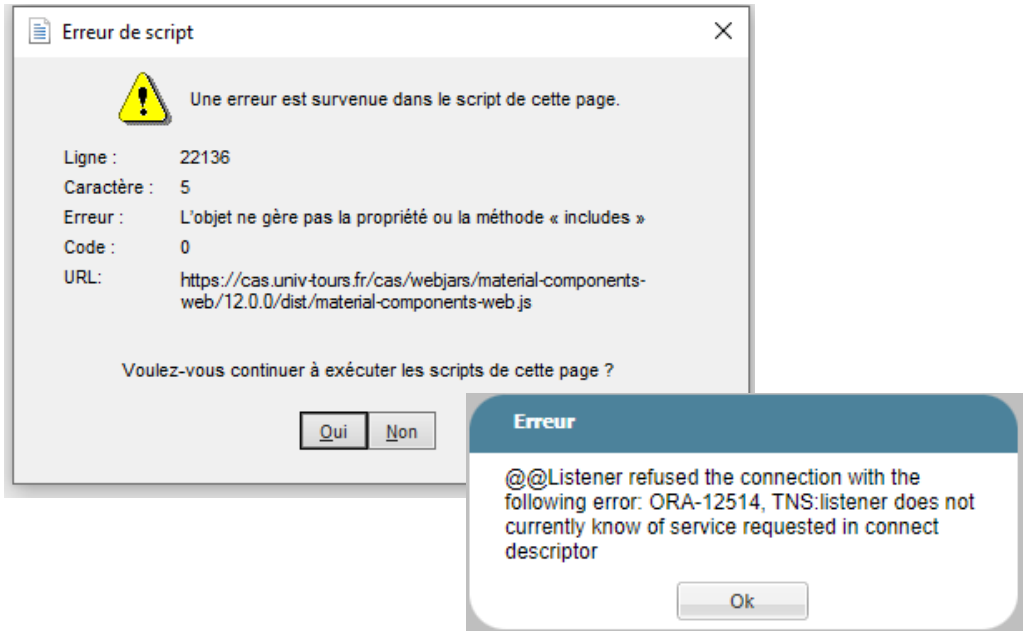
https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

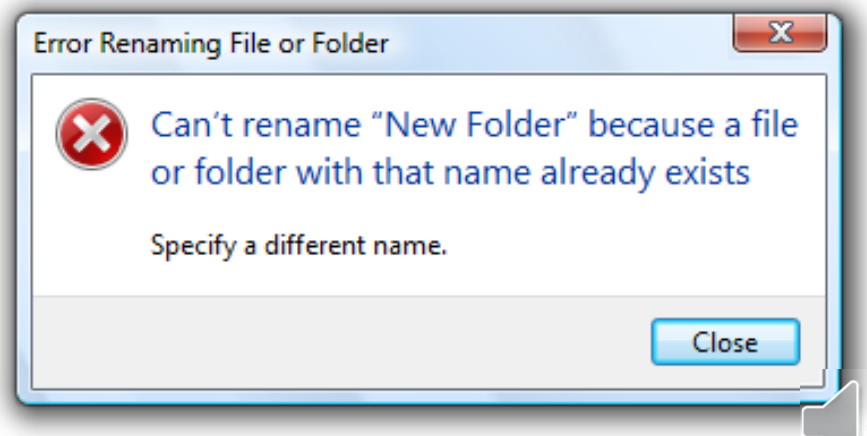
Ben Shneiderman 8 Golden Rules of Interface Design

5 Offer simple error handling

- Be nice when the user is wrong
- Provide explanations about the error
- Tell what to do to solve the problem







https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

Ben Shneiderman 8 Golden Rules of Interface Design

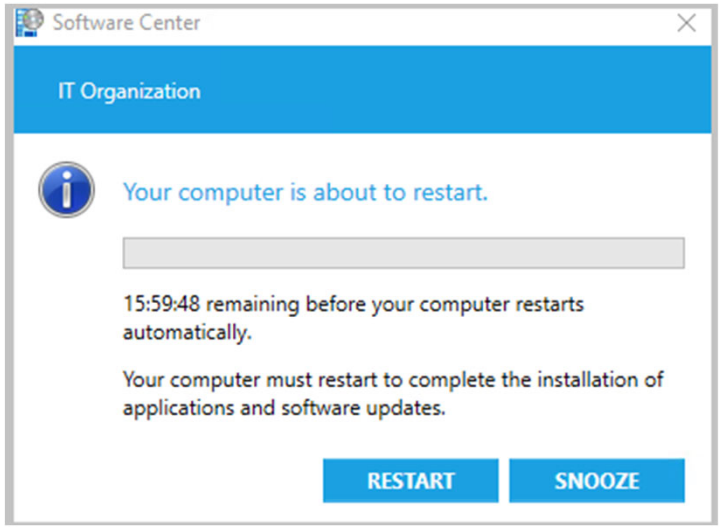
- 6 Permit easy reversal of actions
  - Undo/redo functionality
  - Makes the user feel good (errors can be corrected)

https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

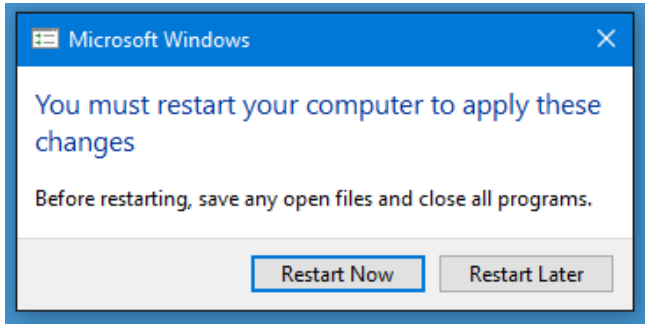
Ben Shneiderman 8 Golden Rules of Interface Design

7 Support “internal locus of control”

- “Control takes place within the user”
- Let the user have control of the GUI and its events
- User should not suffer from the system’s actions



Uncontrolled GUI event



Controlled event

https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

Ben Shneiderman 8 Golden Rules of Interface Design

8 Reduce short-term memory load

- Remember the Miller magic number (7)
- Use visual recognition rather than recall/cognitive load
- Keep the GUI simple

https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

Ben Shneiderman : two additional principles

A pixel is a terrible thing to waste

- Avoid vast area with nothing
- Pack widgets together
- Use adapted windows/elements size

Overview first, zoom and filter, then details-on-demand

- Provide an overall picture of information
- Let the user focus on interesting parts

https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces

Jakob Nielsen's 10 general principles

1. Visibility of system status
   - Provide informative feedback
2. Match between system and the real world
   - Simple vocabulary, concepts close to problem real-world entities
3. User control and freedom
   - Undo/redo, Quit without loss, informative feedback
4. Consistency and standards
   - Consistent design, use established conventions
5. Error prevention
   - Help user avoid errors

https://www.nngroup.com/articles/ten-usability-heuristics/

Jakob Nielsen's 10 general principles

6. Recognition rather than recall
   - Miller magic number, informative icons
7. Flexibility and efficiency of use
   - Shortcuts
8. Aesthetic and minimalist design
   - Focus on relevant information
9. Help users recognize, diagnose, and recover from errors
   - Explain, suggest solutions to errors
10. Help and documentation
    - Contextual help

https://www.nngroup.com/articles/ten-usability-heuristics/
https://media.nngroup.com/media/articles/attachments/Heuristic_Summary1-compressed.pdf

Course on Graphical User Interfaces (GUI)

Session 3: detailed rules for GUI design

Gilles Venturini, venturini@univ-tours.fr, office 204

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

POLYTECH TOURS

université de TOURS

LIFAT EA 6300

Text (labels, input boxes, …)

- Has to be readable
- Size, line spacing , use system font (or standard font), possibly always the same font, left alignment,
- No style, no color, check contrast (text/background)
- AVOID CAPITAL LETTERS, *italic*, underlined

Background (columns) / **Foreground** (rows)

Foreground \ Background	Red	Orange	Yellow	Green	Blue	Violet	Black	White	Gray
Red		Poor	Good	Poor	Poor	Poor	Good	Good	Poor
Orange	Poor		Poor	Poor	Poor	Poor	Good	Poor	Poor
Yellow	Good	Good		Poor	Good	Poor	Good	Poor	Good
Green	Poor	Poor	Poor		Good	Poor	Good	Poor	Good
Blue	Poor	Poor	Good	Good		Poor	Poor	Good	Poor
Violet	Poor	Poor	Good	Poor	Poor		Good	Good	Poor
Black	Poor	Good	Good	Good	Poor	Good		Good	Poor
White	Good	Good	Good	Poor	Good	Good	Good		Good
Gray	Poor	Poor	Good	Good	Poor	Poor	Poor	Good	

Lifewire / Jeremy Girard, https://www.thoughtco.com/contrasting-foreground-background-colors-4061363

- Check size of input boxes

Provide enough space for inp

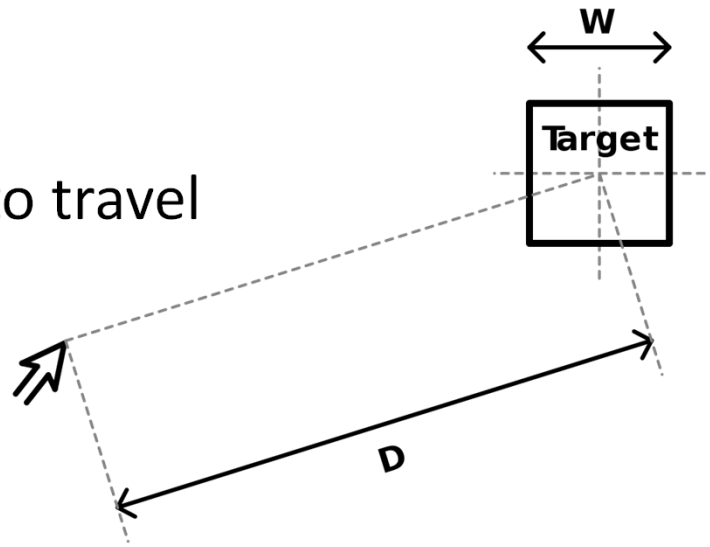
Add ScrollBars for multilines boxes so as to give enough space to the user for text editing, reading, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc,

Message boxes

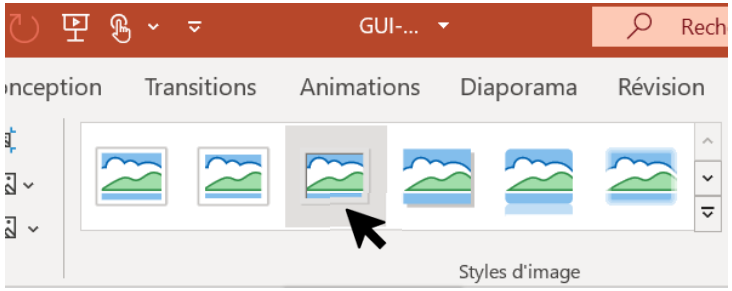
- Give enough time to read: 80 characters -> at least 30 seconds

Buttons

- Time to reach and click depends on distance to travel and button shape/size (Fitts's law)



- Correct size and spacing (+ consistent)
- Clear selection (mouse hovering)



- Use a clear text label that represents the command

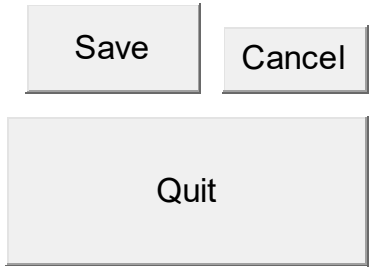
- Avoid symbols unless very clear and shorter than text

https://en.wikipedia.org/wiki/Fitts_law

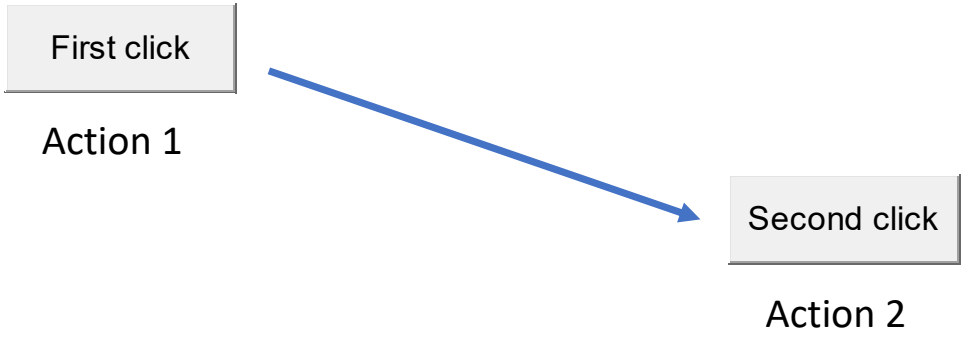
Buttons layout

- Do not place next to each other buttons with opposite actions

- Be consistent and homogeneous



- Minimize travel distance (check scheduling of actions in frequent user's task)



Check boxes and Radio buttons

- Multiple vs exclusive selection

- Vertical alignment
- Clear text labels
- Check Miller's magic number (make several groups if needed)

Clear text explaining context:

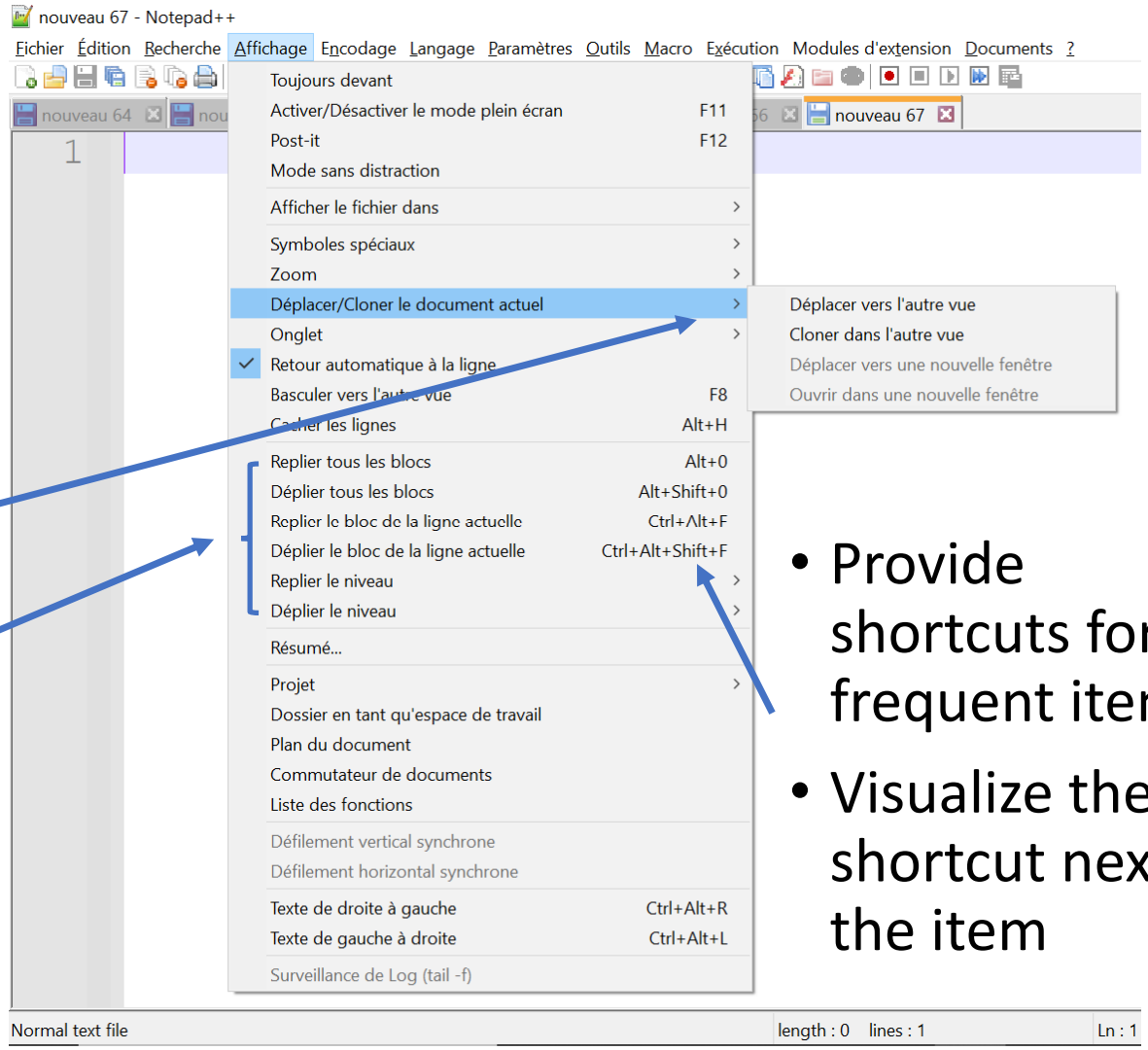
- ☑ Group1-CheckBox1
- ☑ Group1-CheckBox2
- ☐ Group1-CheckBox3

Clear text explaining context:

- ◉ Group2-OptionButton1
- ○ Group2-OptionButton2
- ○ Group2-OptionButton3

Menus

- On top of window (Z-pattern)
- Use clear text labels for menu's items
- Avoid deep hierarchy (1 is ideal, 2 OK, avoid 3)
- Make groups/chunks (Miller magic number)



- Provide shortcuts for frequent items
- Visualize the shortcut next to the item

Breadcrumb trail/Fil d'Ariane

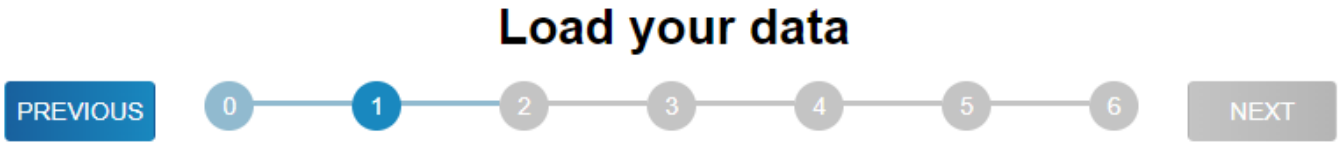
- After a series of actions (navigation/choices), a trail that:
  - Displays the actual path followed/options selected by the user
  - Allows the user to come back to a previous step
  - Helps the user understand the current state of the task being performed

- Examples:
  - Navigation in Celene

https://en.wikipedia.org/wiki/Breadcrumb_navigation
https://fr.wikipedia.org/wiki/Fil_d%27Ariane_(ergonomie)

Breadcrumb trail/Fil d'Ariane

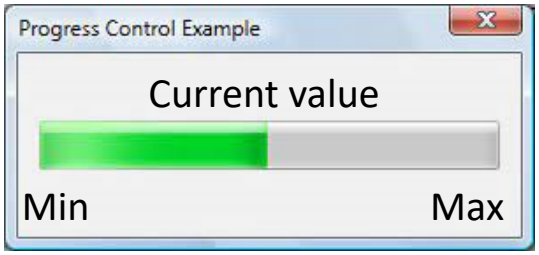
- Examples:
  - Shows the progression of the task being performed (www.vizassist.fr)



  - A path in a tree/hierarchical structure being explored (here windows folders)

Progress bars (loading bars, etc)

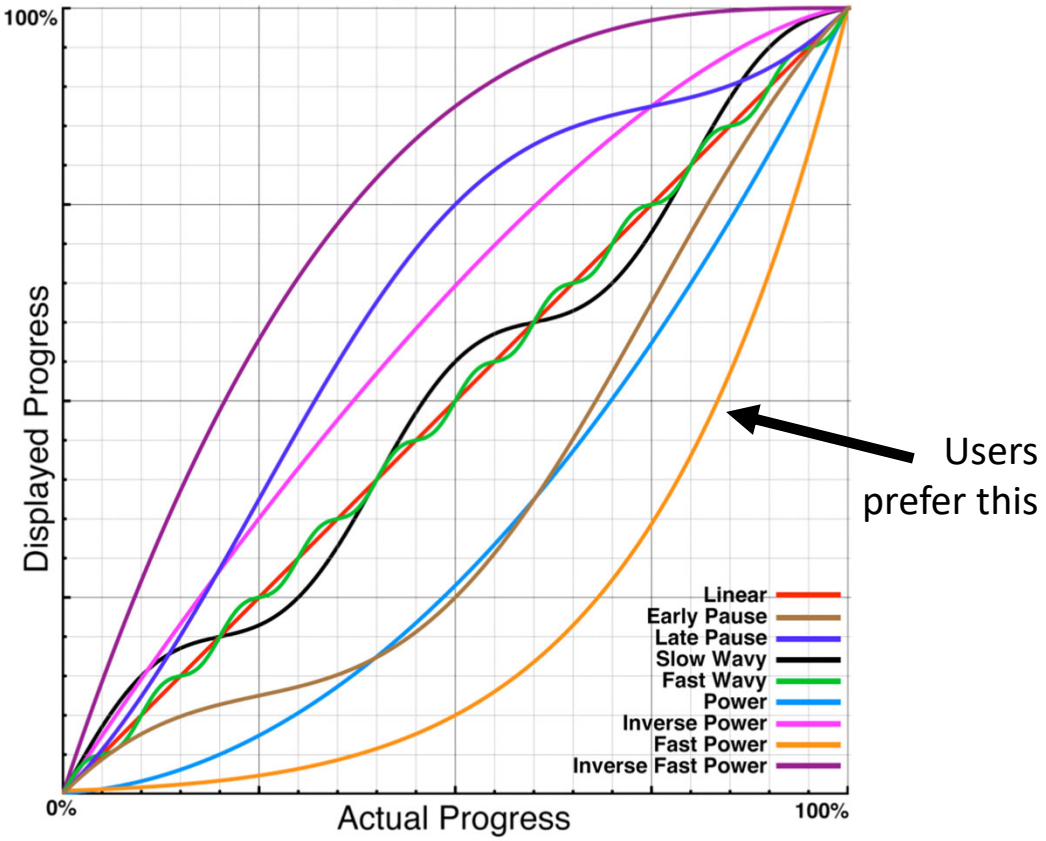
- Helps the user to wait for a result:
  - Shows the progression of a process
  - Gives an indication of the time that remains
- Parameters:



https://en.wikipedia.org/wiki/Progress_bar

Progress bars (loading bars, etc)

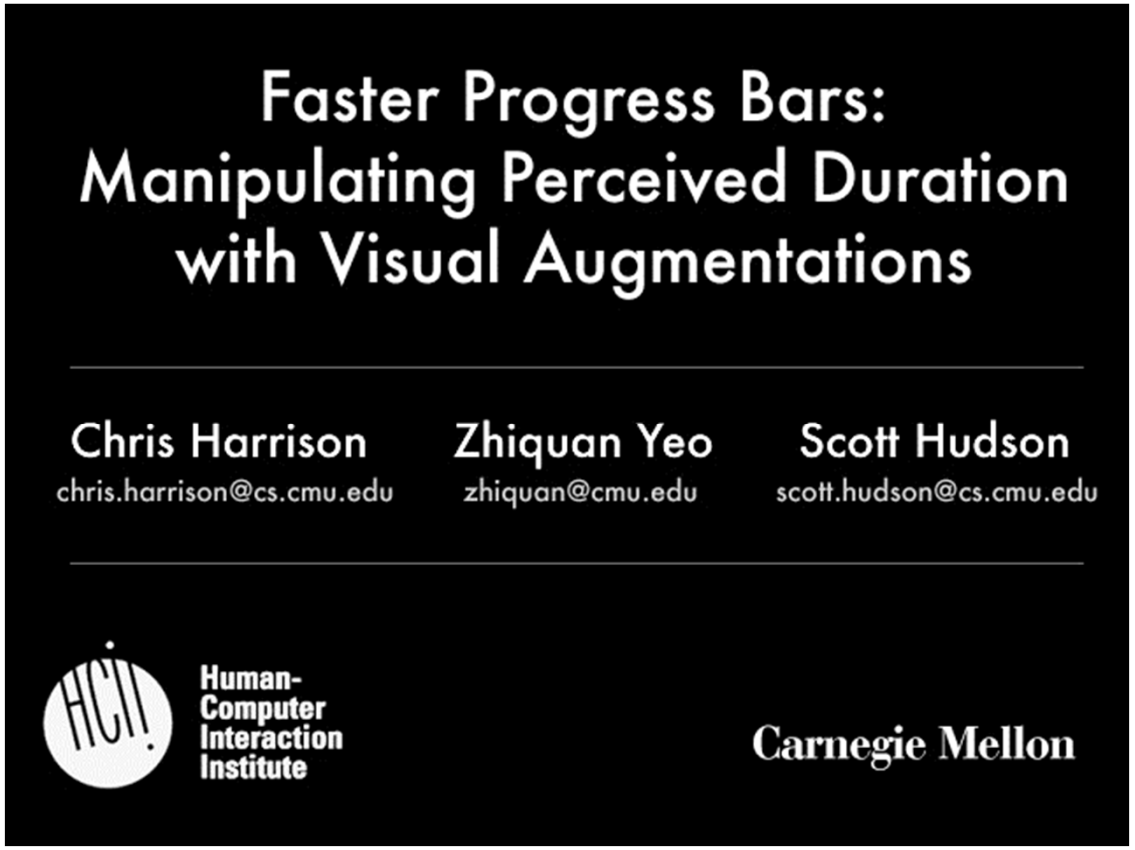
- Threat 1: wrong estimation of time
- Threat 2: Perception of time by humans is non linear
  - So progress bars should be much slower than reality at the beginning,
  - And they should accelerate at the end.



Harrison, C., Amento, B., Kuznetsov, S., & Bell, R. (2007, October). Rethinking the progress bar. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (pp. 115-118).

Progress bars (loading bars, etc)

- Threat 3: other graphical effects can help users wait
- *“visually augmented progress bars could be used to make processes appear 11% faster, when in reality, their duration remains unchanged.”*

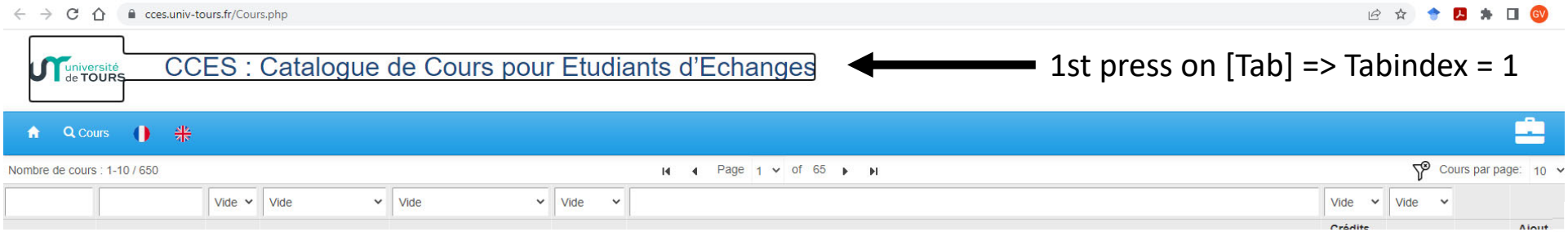


Chris Harrison, Zhiquan Yeo, and Scott E. Hudson. 2010. Faster progress bars: manipulating perceived duration with visual augmentations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '10). Association for Computing Machinery, New York, NY, USA, 1545–1548. https://doi.org/10.1145/1753326.1753556

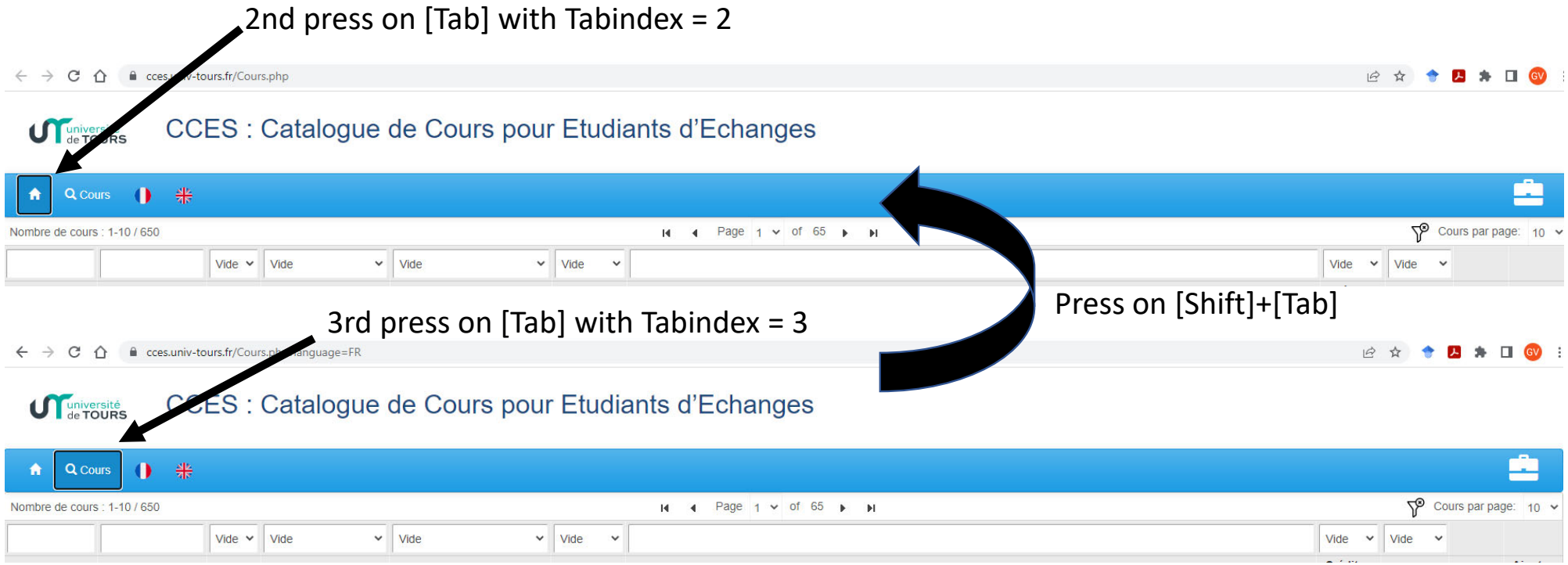
Tabindex property

- The Tabindex property:
  - Many widgets have a tabindex value (buttons, text box, etc)
  - Specify the order in which widgets get the focus when the [Tab] key is pressed (or [Shift]+[Tab] for going back)
  - Speeds up the keyboard interactions

- Example (Chrome):



Tabindex property



- Threat: set Tabindex incorrectly for relevant widgets (and associated task)

Course on Graphical User Interfaces (GUI)

Session 4: implementation with TKINTER

Gilles Venturini, venturini@univ-tours.fr, office 204, 220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

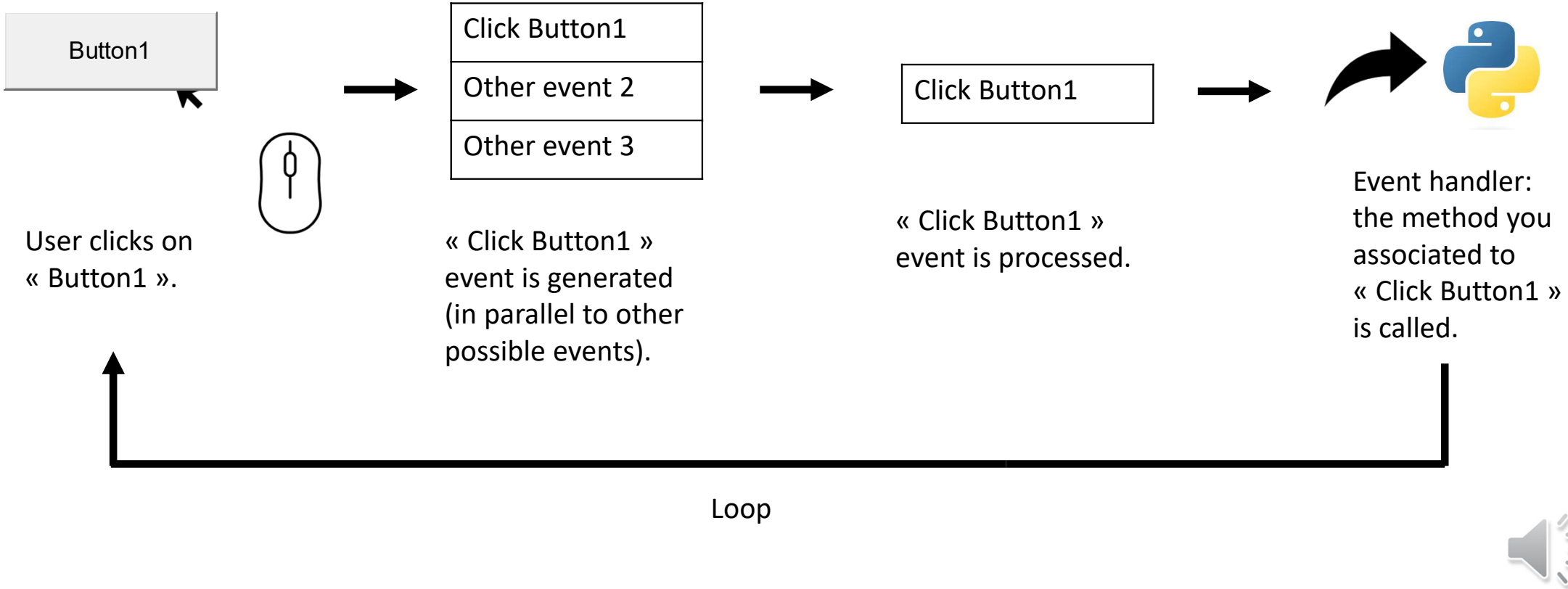
POLYTECH TOURS

université de TOURS

LIFAT EA 6300

GUI event loop

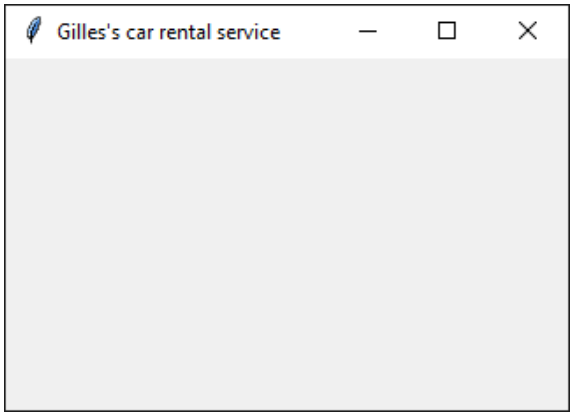
- GUI = event driven program (called GUI manager)



GUI event loop with tkinter

- Launching the event loop of a first window with tkinter:

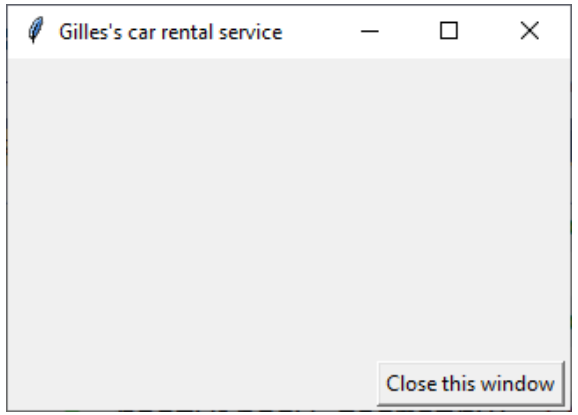
```
# Import the tkinter module
import tkinter as tk

# Create the 320x200 application window
rootwindow = tk.Tk()
rootwindow.geometry("320x200")
rootwindow.title("Gilles's car rental service")

# Start the event loop: the program will remain in
this loop until the window is closed
rootwindow.mainloop()

# So the instruction below will be run only once
the window is closed
print("Loop is finished!")
```



GUI event loop + GUI manager

- Event management is done by GUI manager (not by you)

```
#imports the tkinter module
# Create the application window
…

# Function (event handler) that will be called
when the event "click on button1" is raised
def CommandForButton1():
    # it will close the application window
    rootwindow.quit()

# Add a button to the rootwindow
# 1) define the button, its name, its parent (rootwindow)
and its parameters (text, event handler)
button1 = tk.Button(rootwindow, text="Close this window", command=CommandForButton1)
# 2) define the layout of the button within its parent
button1.place(x=210, y= 172)

# Start the event loop
rootwindow.mainloop()
```



Events properties

- In a GUI, events are generated by:
  - User actions with hardware devices: keyboard (key down, up, [Return], [Tab], etc) and mouse (click, move, etc)
  - And concerning a widget (resizing a window, ticking a box, clicking button, getting the focus, etc)
- Software components can also generate events (timer, etc)
- Events have parameters (additional information) given to event handler
  - Mouse move: X,Y coodinates

Binding widgets and events

- In previous code:

```
# Add a button to the rootwindow
# 1) define the button, its name, its parent (rootwindow)
and its parameters (text, event handler)
button1 = tk.Button(rootwindow, text="Close this window", command=CommandForButton1)
```

- Events can be bind to a widget:

```
# Consider a button
button1 = tk.Button(rootwindow, text="Close this window")
# You can bind many events to this button
button1.bind('<Button-1>',EventHandlerForLeftClick)
button1.bind('<Button-3>',EventHandlerForRightClick)
```

Binding widgets and events

- What events are defined? (some can be specific to a widget):

Category	Instance	Role
<Button>	<Button-1>, <Button-2>, <Button-3>	Left, middle or right click with the mouse
<ButtonRelease>	<ButtonRelease-1>, <ButtonRelease-2>, <ButtonRelease-3>	Release of left, middle, or right button
<Double-Button>	<Double-Button-1>, <Double-Button-2>, …	Double click …
<Enter>, <Leave>	<Enter>, <Leave>	Mouse pointer enters/leaves the widget
<FocusIn>,<FocusOut>	<FocusIn>, <FocusOut>	Widget obtains/loses keyboard focus
…	…	…

- It is difficult to find a list of events for tkinter (see following links)

https://www.tcl.tk/man/tcl8.5/TkCmd/bind.html
https://tkdocs.com/shipman/event-types.html

GUI event loop: frozen interface

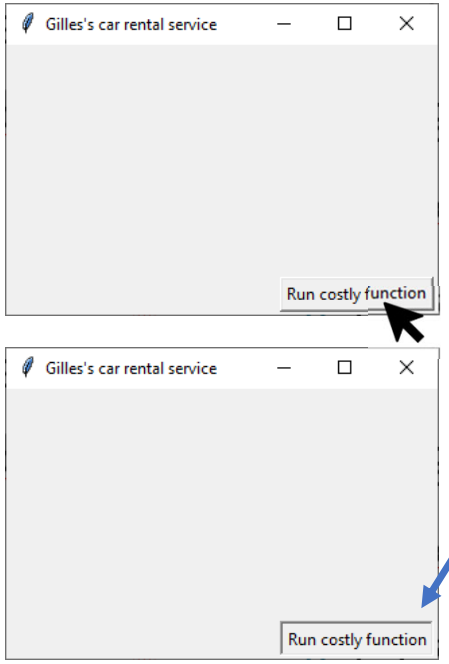
- With tkinter, GUI manager + your main program run on the same thread
- GUI might be « frozen » if high computation is required by your program

Function that takes several seconds to run

```
def HighComputationalCost():
    maxrange = 200
    for i in range(0,maxrange):
        for j in range(0,maxrange):
            for k in range(0,maxrange):
                sum = i + j + k
    print("Done computation.")
```

Will be the event handler for « click on button1 »

```
# Event handler for "click on button1"
def CommandForButton1():
    # function with a high computational cost
    HighComputationalCost()
    print("Done button1 event handling.")
```



Frozen interface

```
Done computation.
Done button1 event handling.
```

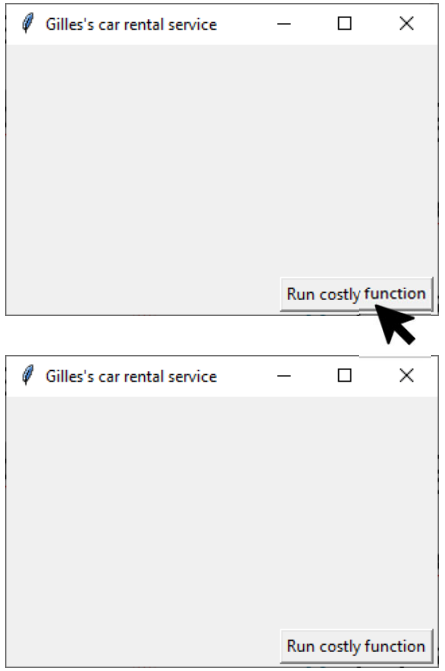
Console output

GUI event loop: frozen interface

- Solution: run GUI manager + your function on a different thread

```
# Threads module
import threading

# Event handler for "click on button1"
def CommandForButton1():
    threading.Thread(target=HighComputationalCost).start()
    print("Done button1 event handling.")
```

Function is called in a different thread (in parallel)



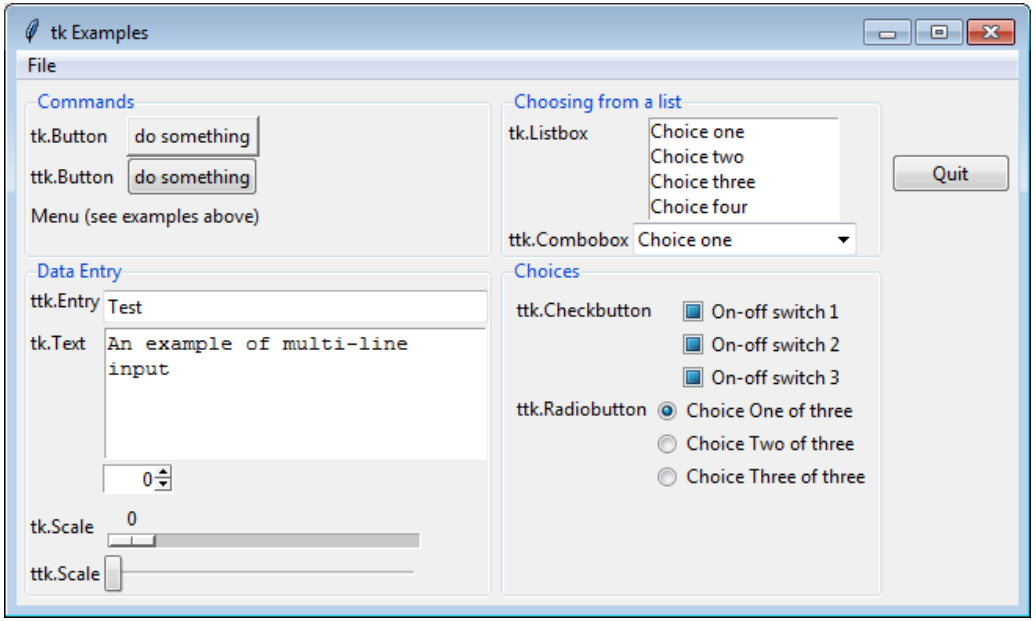
Interface is not frozen

```
Done button1 event handling.
Done computation.
```

Console output

Standard widgets in tkinter

- tk (initial version), ttk (more recent, better design, more widgets)
- tk,ttk: *Button, Checkbutton, Entry, Frame, Label, LabelFrame, Menubutton, PanedWindow, Radiobutton, Scale, Scrollbar, Spinbox*
- ttk: + *Combobox, Notebook, Progressbar, Separator, Sizegrip, Treeview and others*
- Tix, but deprecated since version 3.6



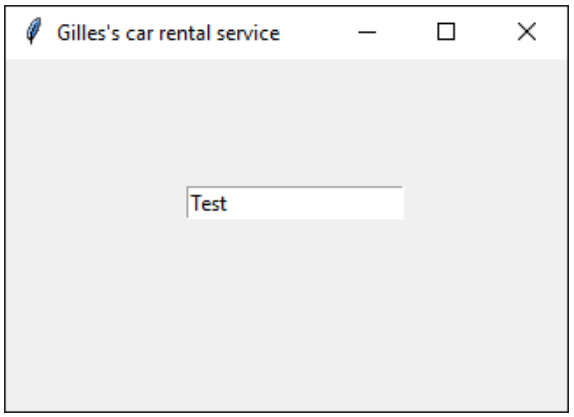
https://docs.python.org/3/library/tkinter.ttk.html
https://runestone.academy/ns/books/published/thinkcspy/GUIandEventDrivenProgramming/03_widgets.html
https://docs.python.org/3/library/tkinter.tix.html

Text entry

- Define the widget and the input text variable

```
# Imports the tkinter module
…
# Creating the variable that will contain the input text,
# with tk special class for strings ...
input_text = tk.StringVar()

#creating the entry widget
entry1 = tk.Entry(rootwindow, textvariable = input_text)
entry1.place(x=103, y=72)

# Start the event loop
rootwindow.mainloop()

# Print the result in console
print(input_text.get())
```

Tells tk that the input text is stored in the « input_text » variable




```
Test
Press any key to continue . . .
```

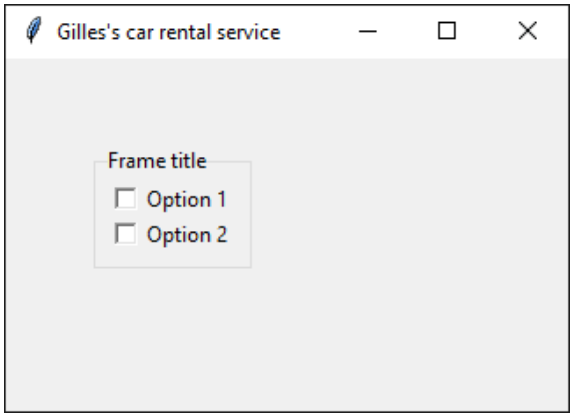
Console output

https://runestone.academy/ns/books/published/thinkcspy/GUIandEventDrivenProgramming/03_widgets.html

LabelFrame

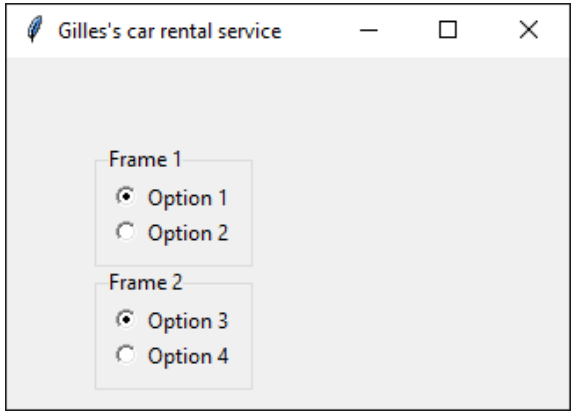
- A structuring widget to cluster together other widgets

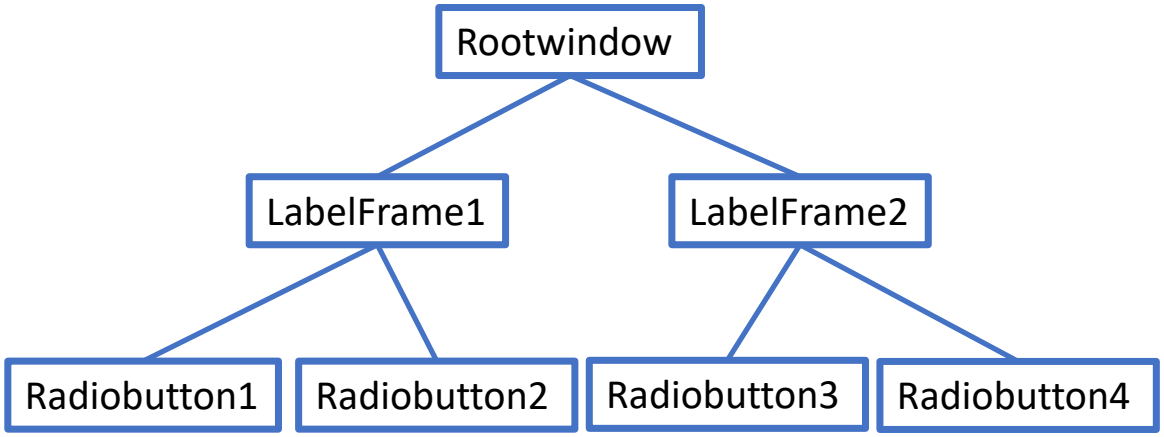
```
# Import ttk
import tkinter.ttk as ttk

# Create the label frame
frame1 = ttk.Labelframe(rootwindow,text='Frame title’,height="70",width="90")
# Define the layout
frame1.place(x=50, y=50)

# Create content attached to the frame
# Two check buttons
checkbutton1 = tk.Checkbutton(frame1,text='Option 1')
checkbutton1.place(x=5, y=0)
checkbutton2 = tk.Checkbutton(frame1,text='Option 2')
checkbutton2.place(x=5, y=20)

# Start the event loop
rootwindow.mainloop()
```



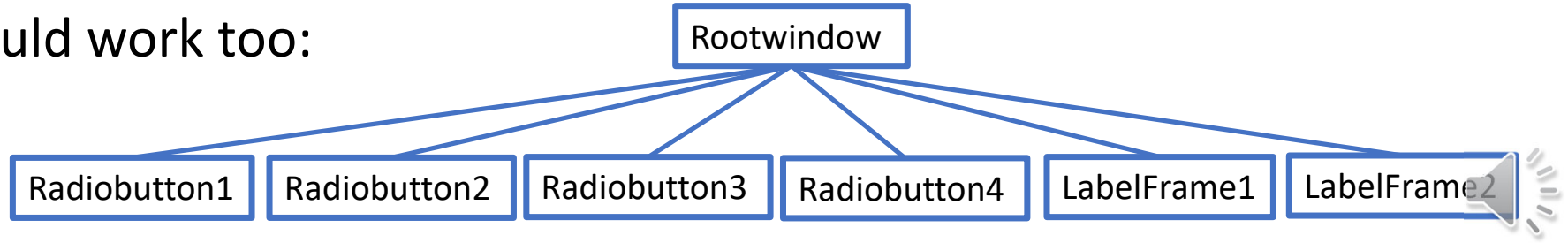
Widgets hierarchy

- Widget can be organized in a hierarchy (defined by you)
- Each widget has a parent (except the root window)





- But this could work too:



Widgets hierarchy

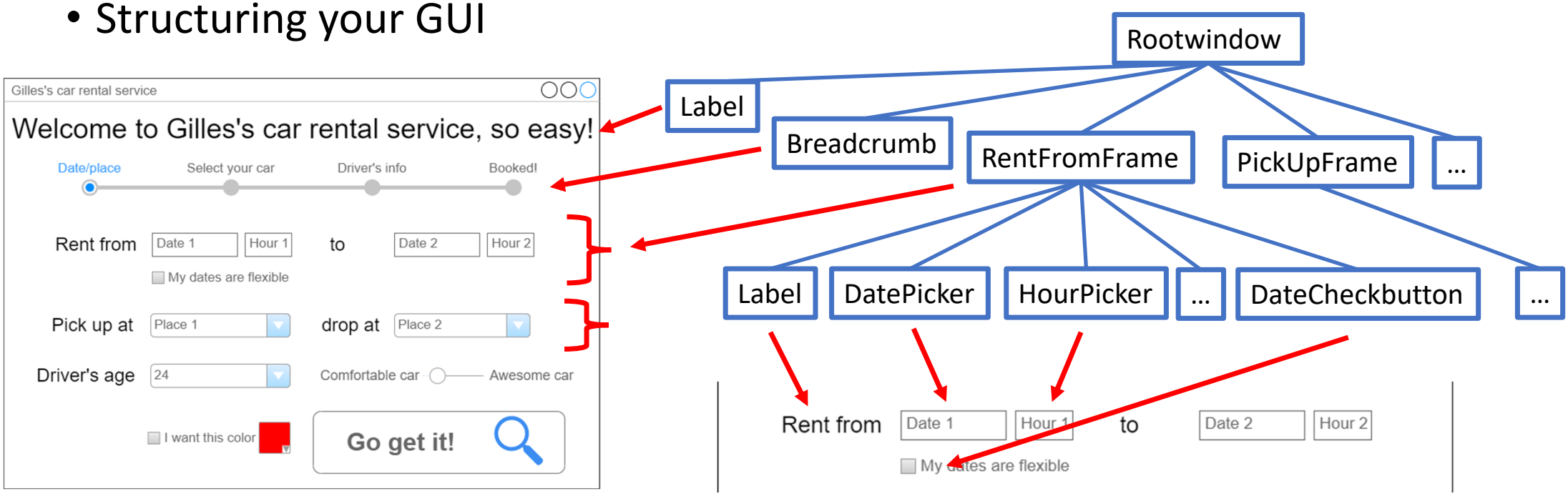
The parent of the widget

```
# Create frame 1
frame1 = ttk.Labelframe(rootwindow,text='Frame 1',height="70",width="90")
# And its layout
frame1.place(x=50, y=50)

# Create two check buttons attached to frame 1
opt12 = tk.StringVar() # tk variable for radio button
opt12.set("Option 1") # Default values for radio buttons
checkbutton1 = tk.Radiobutton(frame1,text='Option 1',variable=opt12,value="Option 1")
checkbutton1.place(x=5, y=0)
checkbutton2 = tk.Radiobutton(frame1,text='Option 2',variable=opt12,value="Option 2")
checkbutton2.place(x=5, y=20)

# Create frame 2
frame2 = ttk.Labelframe(rootwindow,text='Frame 2',height="70",width="90")
# And its layout
frame2.place(x=50, y=120)

# Create two check buttons attached to frame 2
opt34 = tk.StringVar()
opt34.set("Option 3") # Default values for radio buttons
checkbutton3 = tk.Radiobutton(frame2,text='Option 3',variable=opt34,value="Option 3")
checkbutton3.place(x=5, y=0)
checkbutton4 = tk.Radiobutton(frame2,text='Option 4',variable=opt34,value="Option 4")
checkbutton4.place(x=5, y=20)
```

Coordinates within the parent widget

Widgets hierarchy

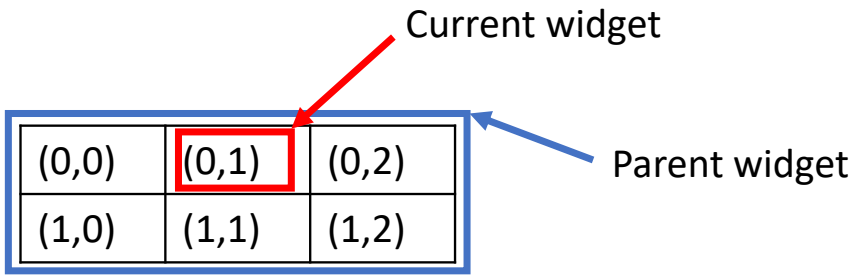
- Structuring your GUI



Widgets layout

3 methods:

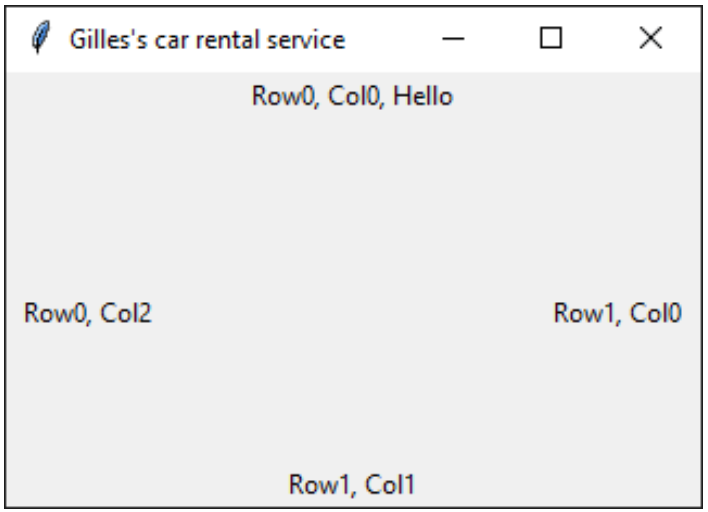
- .place(x =…,y=…): see the previous examples, you set the X and Y coordinates within the parent widget. Can use relx and rely (relative coordinates like 0.25, 0.5)
- .pack(), pack(side=tk.TOP): add automatically the widgets on TOP, LEFT, RIGHT, BOTTOM side of the parent widget in the specified order
- .grid(row=…, column=…)



Widgets layout

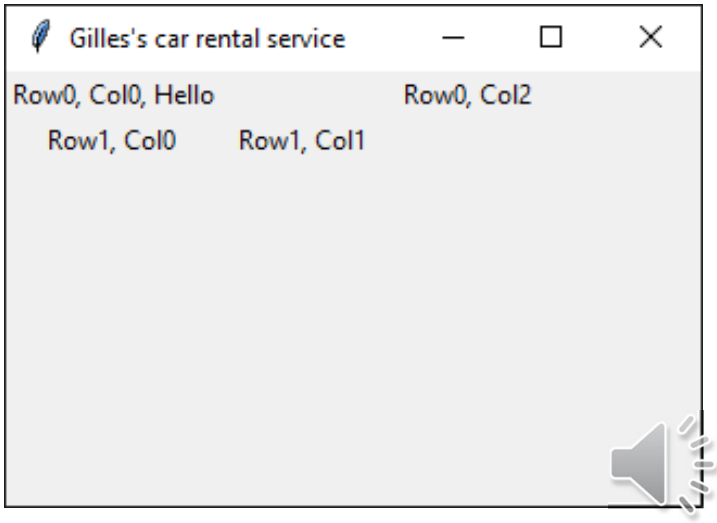
- .pack(),

```
tk.Label(text="Row0, Col0, Hello").pack(side=tk.TOP)
tk.Label(text="Row0, Col2", width=10).pack(side=tk.LEFT)
tk.Label(text="Row1, Col0", width=10).pack(side=tk.RIGHT)
tk.Label(text="Row1, Col1", width=10).pack(side=tk.BOTTOM)
```



- .grid(row=..., column=...)

```
tk.Label(text="Row0, Col0, Hello").grid(row=0, column=0)
tk.Label(text="Row0, Col2", width=10).grid(row=0, column=2)
tk.Label(text="Row1, Col0", width=10).grid(row=1, column=0)
tk.Label(text="Row1, Col1", width=10).grid(row=1, column=1)
```



Widgets state

- Widget are organized in a hierarchy
- Each widget has a parent (except the root window)

???

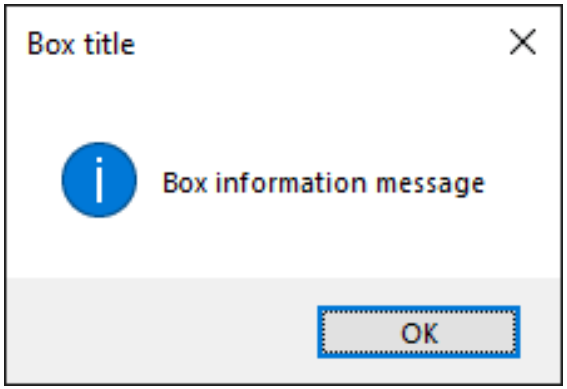
Breadcrumbs ..

- Well, … I found no component for this with tkinter
- When you design your interface, remember about technological constraints.

Dialog boxes

- Simple message box

```
# Imports the tkinter module
…

# Import specific module
import tkinter.messagebox as tkmsgbox

# Shows the messagebox
tkmsgbox.showinfo("Box title", "Box information message")

# Program is stopped until box validation

# Start the event loop
rootwindow.mainloop()
```

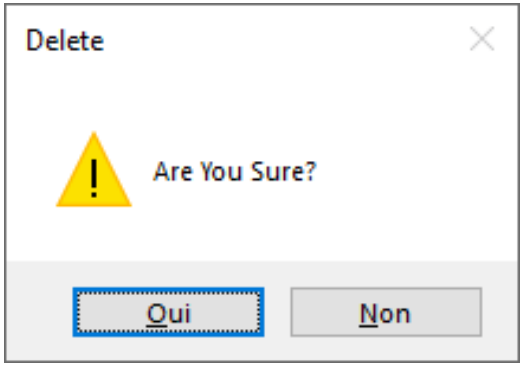


https://docs.python.org/3/library/tkinter.messagebox.html

Dialog boxes

- Message box with question

```
# Import the tkinter module + specific module
…
# Show the messagebox
response = tkmsgbox.askquestion("Delete", "Are You Sure?", icon='warning')
if response == 'yes':
    print("Deleted")
else:
    print("Not Deleted")

# Start the event loop
rootwindow.mainloop()
```



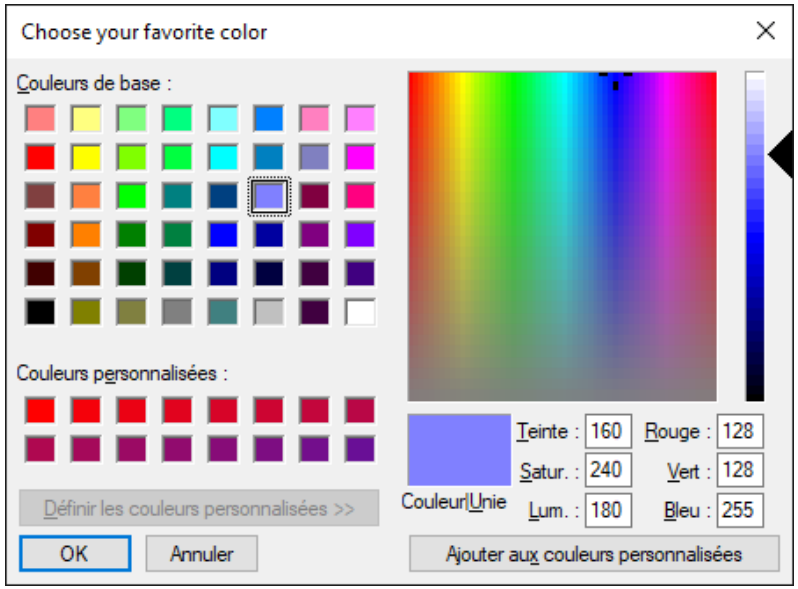
- Several other boxes: askokcancel, askretrycancel, …

https://docs.python.org/3/library/tkinter.messagebox.html

Dialog boxes

- Color chooser

```
# Import the tkinter module + specific module
…
# Import specific package
import tkinter.colorchooser as tkcolchooser

# Show the dialog box
color= tkcolchooser.askcolor(title ="Choose your
favorite color")
print(color)

# Start the event loop
rootwindow.mainloop()
```




```
((128.5, 128.5, 255.99609375), '#8080ff')
```

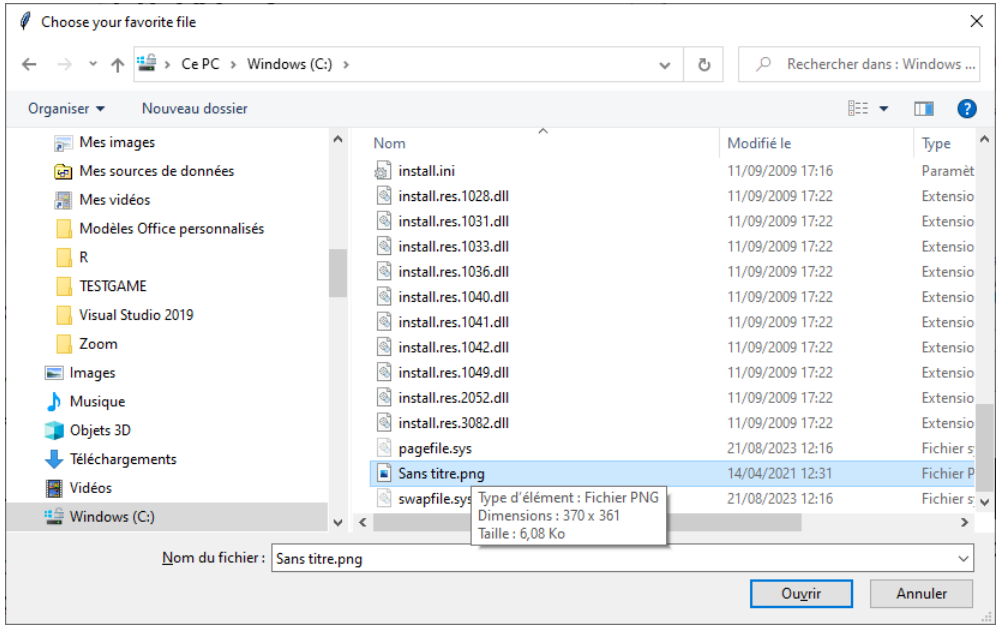
Console output (RGB encoding)

https://docs.python.org/3/library/tkinter.colorchooser.html

Dialog boxes

- Open file dialog

```
# Import the tkinter module + specific module
…
# Import specific module
import tkinter.filedialog  as tkfiledial

# Show the dialog box
filename= tkfiledial.askopenfilename(title =
"Choose your favorite file")
print(filename)

# Start the event loop
rootwindow.mainloop()
```




```
C:/Sans titre.png
```

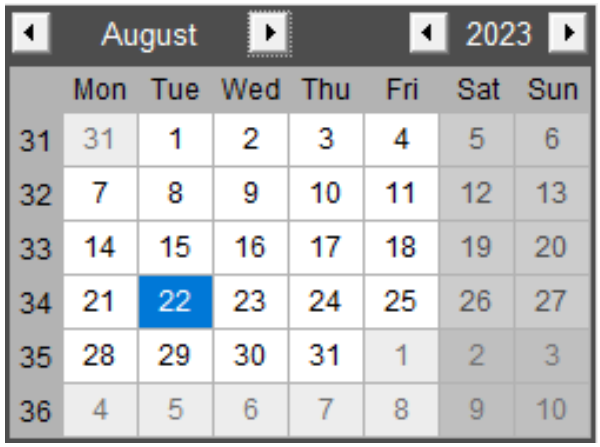
Console output

- Other such dialogs = multiple files, save as, directory, …

https://docs.python.org/3/library/dialog.html

Dialog boxes from other librairies

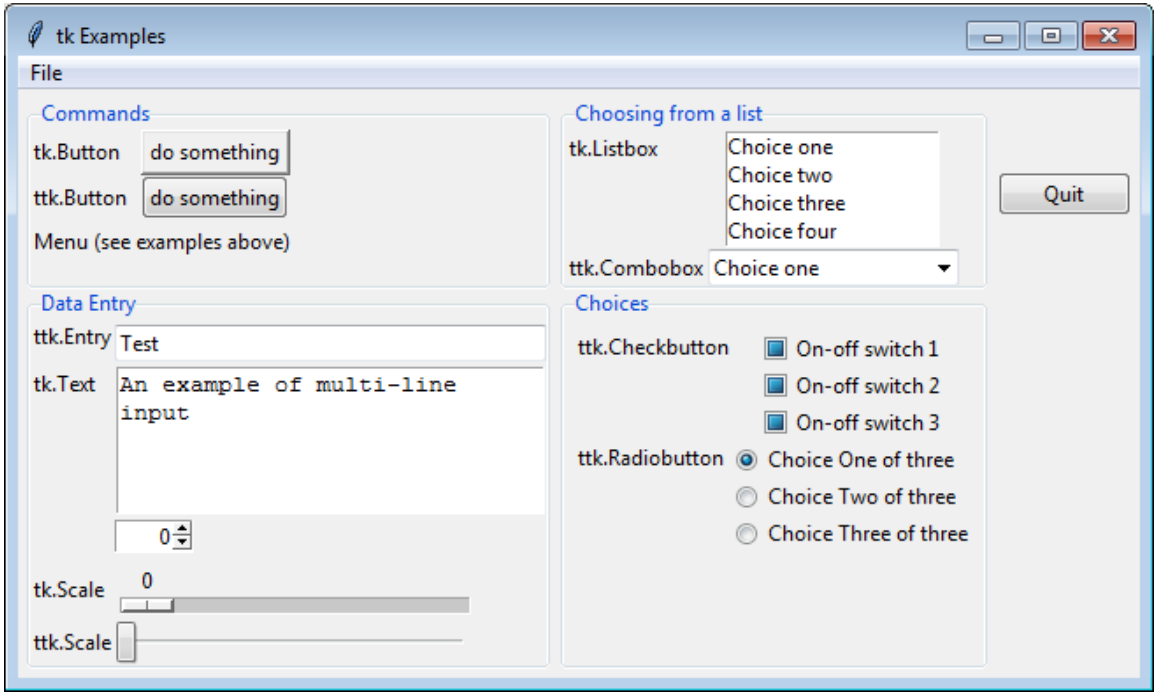
- Date picker



https://www.geeksforgeeks.org/create-a-date-picker-calendar-tkinter/

More windows

- tk and ttk …
- Most important widgets

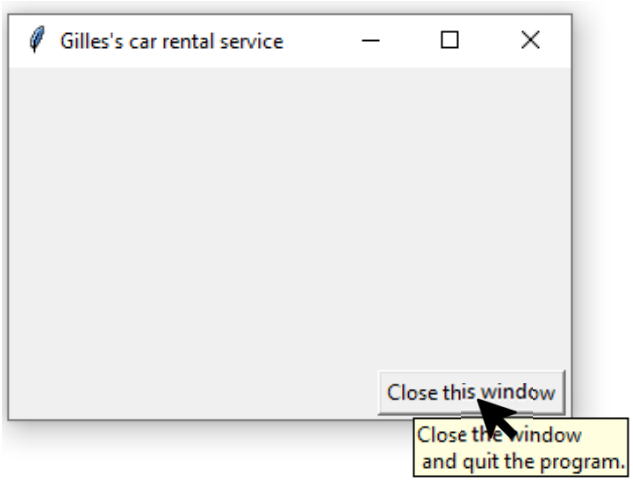


https://runestone.academy/ns/books/published/thinkcspy/GUIandEventDrivenProgramming/03_widgets.html

Tool tips

- Important to help users
- Not directly with tk, but additional library
- Example below: adding a tooltip to « button1 »

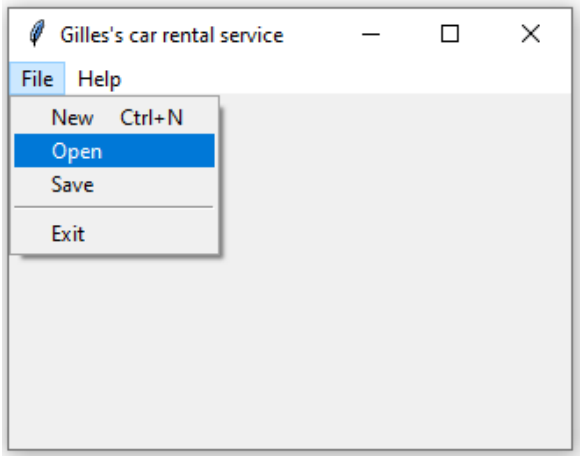
```
# Import specific module
import idlelib.tooltip  as tltip

# Define the tooltip
myTip = tltip.Hovertip(button1,'Close the window \n
and quit the program.', hover_delay=500)
```



https://runestone.academy/ns/books/published/thinkcspy/GUIandEventDrivenProgramming/03_widgets.html

Menus

- A menu belongs to the menubar
- Menu « File » is a kind of list of items
- Items can have shortcuts (requires event binding)
- Items can be grouped with separators

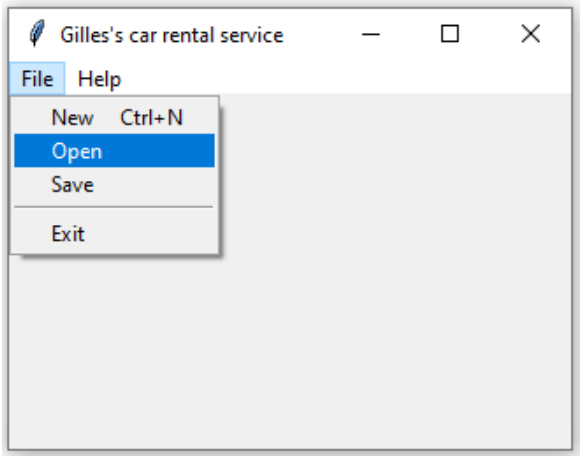



```
# A function (event handler)
def myAction(event=""):
    print("done")

# Menu belongs to menubar created below
menubar = tk.Menu(rootwindow)
```

https://runestone.academy/ns/books/published/thinkcspy/GUIandEventDrivenProgramming/03_widgets.html

Menus

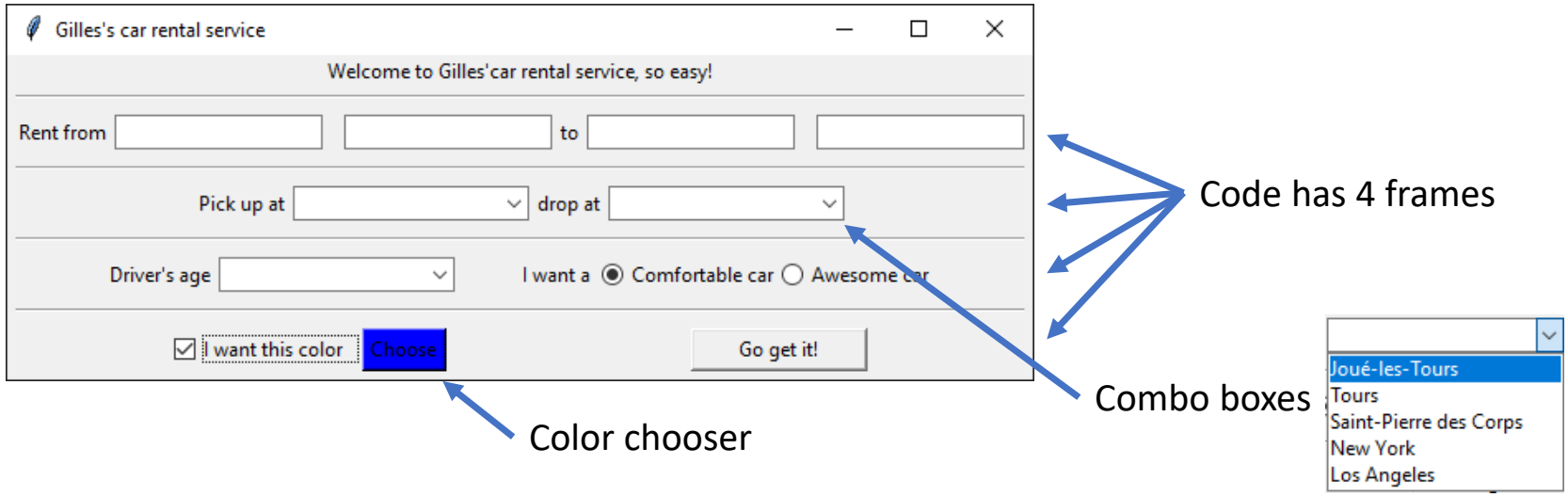
```
# Definition of a file menu
filemenu = tk.Menu(menubar, tearoff=0)

# Add commands to this menu
filemenu.add_command(label="New", command=myAction, accelerator= "Ctrl+N") # to show shortcuts
filemenu.add_command(label="Open", command=myAction)
filemenu.add_command(label="Save", command=myAction)
filemenu.add_separator() # to make groups of commands
filemenu.add_command(label="Exit", command=rootwindow.quit)

# Adding this menu to the menubar
menubar.add_cascade(label="File", menu=filemenu)

# A second part of the menu
…

# Attach the menubar to the main window
rootwindow.config(menu=menubar)

# Create binding for shortcut
rootwindow.bind("<Control-n>", myAction)
```



https://runestone.academy/ns/books/published/thinkcspy/GUIandEventDrivenProgramming/03_widgets.html

First prototype

- Here a first prototype for « Gilles's car rental service »



- Needs a date + hour picker, better decoration, layout and interaction …

- All code examples are on celene.univ-tours.fr

Course on Graphical User Interfaces (GUI)

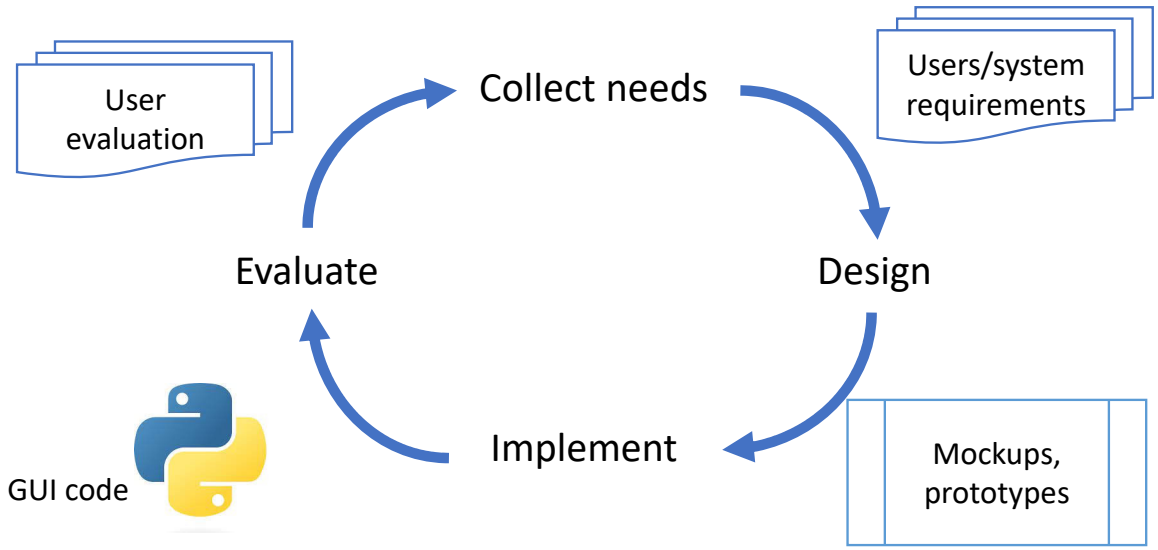
Session 5: UX, Usability and User Evaluation

Gilles Venturini, venturini@univ-tours.fr, office 204

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637

Remember GUI conception

- GUI development cycle (partial or complete GUI):



- After implementation: back to users!

- Testing, evaluating your GUI … can lead to new requirements and design

UX (User Experience)

- UX = everything about user interactions with your GUI, from the beginning to the end:
  - Feelings, perceptions, opinions, thoughts about your GUI
  - GUI is fun, nice, ugly, beautiful, fast, slow, efficient, useful, useless, annoying, great, easy, difficult, messy, cute, pleasant, rotten (and even less polite words …), etc.
- UX is highly subjective, irrational which can lead to situations like this:

GUI A	GUI B	
User not so happy	User highly happy	*Users prefer interface B!*
Highest performances (shorter time to solutions, better solutions)	Medium performances	

- UX can only be understood with final users (not you and developpers, etc) and real experiments called user evaluations

https://en.wikipedia.org/wiki/User_experience

Usability

- Concept close to UX but with a more quantitative/pragmatic point of view
- Usability = Learnability (how fast user can learn the GUI), Efficiency (for task solving), Memorability (using after a long break), Errors (how many, critical or not, recovery), Satisfaction
- Can be measured with objective criteria
- Experiments to evaluate usability = user evaluations

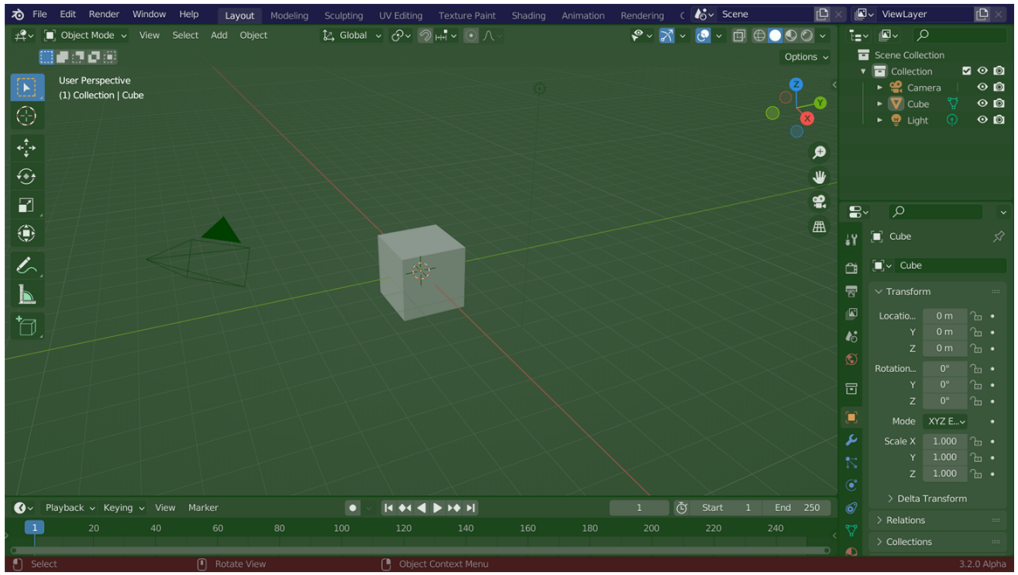
Threat (with negative UX or low usability):

- user will give up (leaving the web site, un-installing the app, etc)
- user will solve the problem with difficulties => low business efficiency
- dramatic consequences (https://fr.wikipedia.org/wiki/Catastrophe_du_mont_Sainte-Odile)

https://en.wikipedia.org/wiki/Usability

Learnability

- User require a (long) learning time for your GUI
  - Example, consider two following (FL studio, Blender):



  - Initially, the user is not familiar with your GUI
  - But the user learns and progresses in its ability to use the interface and solve tasks

Learnability

- Threat (if learning time is too long):
  - Might introduce a bias in a user evaluation: the user must have enough time to learn your GUI or the measured performances will be low whatever the interface => propose a training period

- How to minimize learning time
  - Give enough time and be patient
  - Intuitive and efficient design/conception: use expected standards, follow perception and conception rules, … (many points studied in this course)
  - Provide help within the interface (especially at the first launches) or tutorials

Accessibility

- Visually impaired people, children, elders
- Can they use your interface?

- Close your eyes, and test your GUI with a screen reader like NVDA (https://www.nvaccess.org/)

- Screen reader compatible with tkinter? (but possible with wxPython)

User evaluation

- Two interfaces for Gilles's car rental service: GUI A, GUI B
- Which one is better?
- Or suppose you want to evaluate your interface

$\Rightarrow$ Design a user evaluation

- Define the protocol (pool of testers, forms to fill in, tasks to solve, evaluation measures)
- Do a pilot study with a few testers (to be sure your protocol is valid)
- Perform the evaluation
- Analyze results and take a decision, A or B!

Task-based user evaluation protocol

- Recruit target users (enough for statistical tests)
- Define the aparatus (computer power, screen resolution, etc)
- Prepare a schedule for 1 tester
  - Participant welcome, 5 minutes
  - Fill a pre-form (user characteristics: age, …, level in CS, …), 5 minutes
  - Explain context + GUI + the tasks to be solved, 10 minutes
  - Let the user train with your GUI (learning times), 10 minutes
  - Let the user solve the tasks (and do measurements: time to solve, quality of solution, store behavior in log, camera, …), 30 minutes
  - Fill a final questionnaire (UX oriented, you prefer A or B, etc), 5 minutes
  - Provide a small reward (announce it a the beginning! Candies, cash etc)

  => Already 1H for each tester!

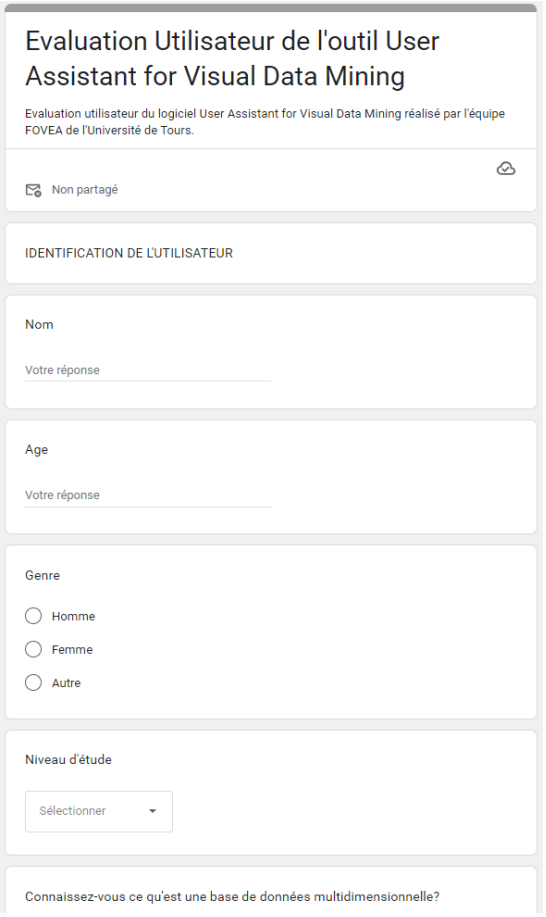
Example of tasks

$\Rightarrow$ Renting a car from Tours to Paris, on the 10th of december, possibly a red car, with four passengers, etc

- Measure time to accomplish the task
- Measure quality of solution found by user (correctness, errors, …)

An example of a real user evaluation

- https://docs.google.com/forms/d/e/1FAIpQLScjLDP-SaXqn9JLdNfXbnIotP1UCyMTcLniMGRR2RbclNg45Q/viewform?usp=sf_link
- Comparative study of 2 GUIs
- Task-based procotol
- Pre-questionnaire, learning phase, tasks solving, post-questionnaire



Compare GUI A and GUI B (task-based)

- Make statistical tests (t-test, etc)

Users	Tasks	GUI A (time)	GUI B (time)	GUI A (quality)	GUI B (quality)
1	T1	2s	4s	80 %	60%
2	T1	3s	5s	70 %	50%
1	T2				
2	T2				
…					

https://biostatgv.sentiweb.fr/ https://med.univ-tours.fr/medias/fichier/biostatgv_1655806847946-pdf

Other points of view (on GUI evaluation)

- Early interface review (with specialists): check the principles presented earlier
- Think aloud protocol: ask the user to speak during the use of the interface (what is the user doing, expecting, what difficulties etc). Small number of users, but highly informative. Requires recording and post-analysis

https://www.boldare.com/blog/what-is-a-thinking-aloud-protocol/

Course on Graphical User Interfaces (GUI)

End of the course!

Gilles Venturini, venturini@univ-tours.fr, office 204, 220

On Celene: https://celene.univ-tours.fr/course/view.php?id=16637